Lancer de rayons et anti-aliasing

David Roger

Ph.D. Student ARTIS
david.roger@inrialpes.fr

Plan

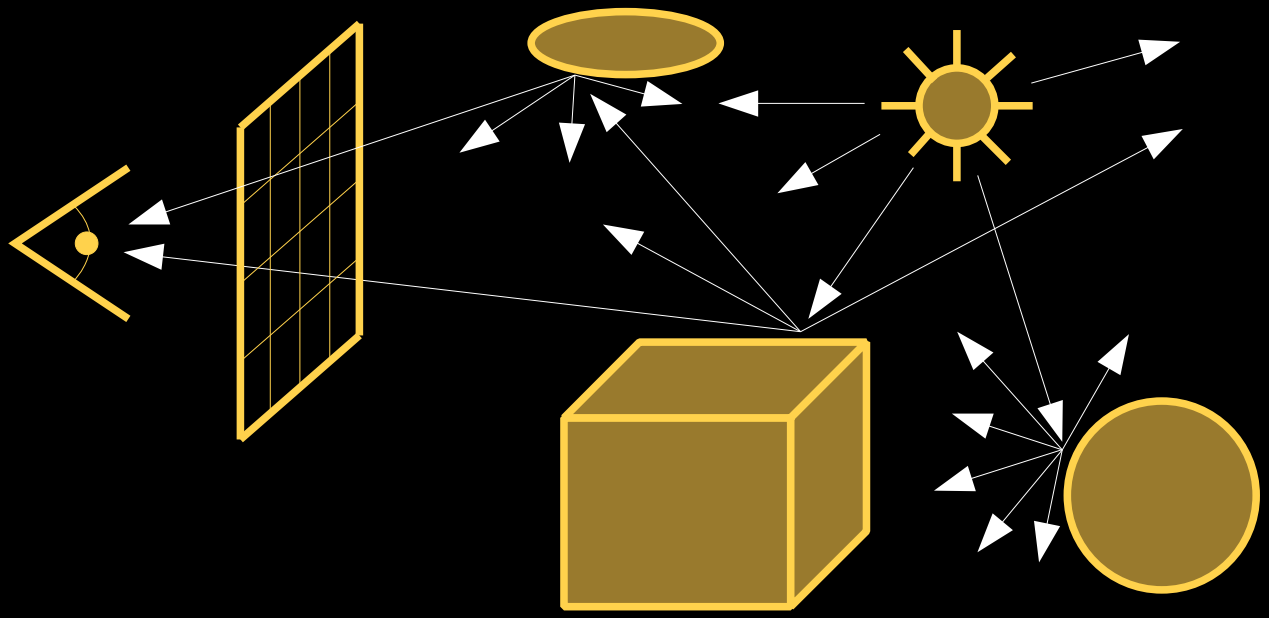
- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Contexte

- Technique de rendu d'images
  - Entrée : description de scène 3D
  - Sortie : image (2D)
- Très générale
- Propagation de rayons lumineux
  - Lois de l'optique géométrique
- Applications dans d'autres domaines
  - Calculs de visibilité
  - Brique d'autres algorithmes de rendu plus complexes

Propagation de la lumière

- Vue comme un ensemble (infini) de rayons
  - Partant de la source
  - Diffusés, réfléchis, transmis ...



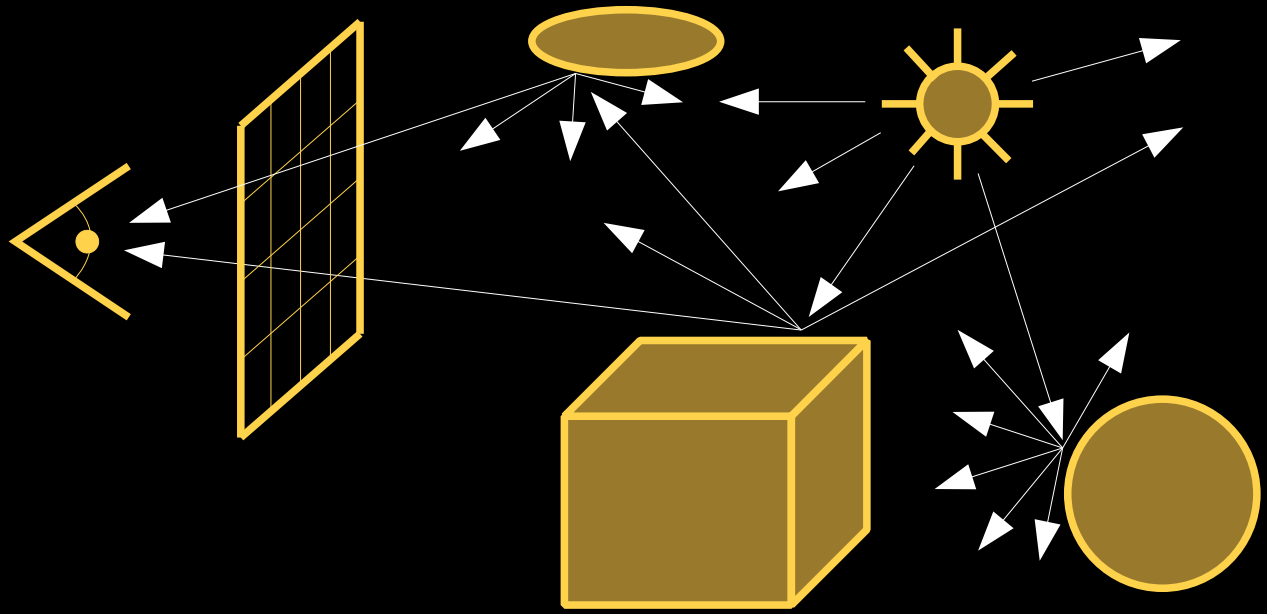
Exemple de simulation

Simulation directe

- Echantillonage de l'ensemble de rayons
  - Approximation : on ne garde qu'un nombre fini de rayons
  - Calcul des trajectoires des rayons
  - Très grand nombre de rayons : méthode (très) lente
- Très réaliste
- Infaisable en pratique

Simulation directe

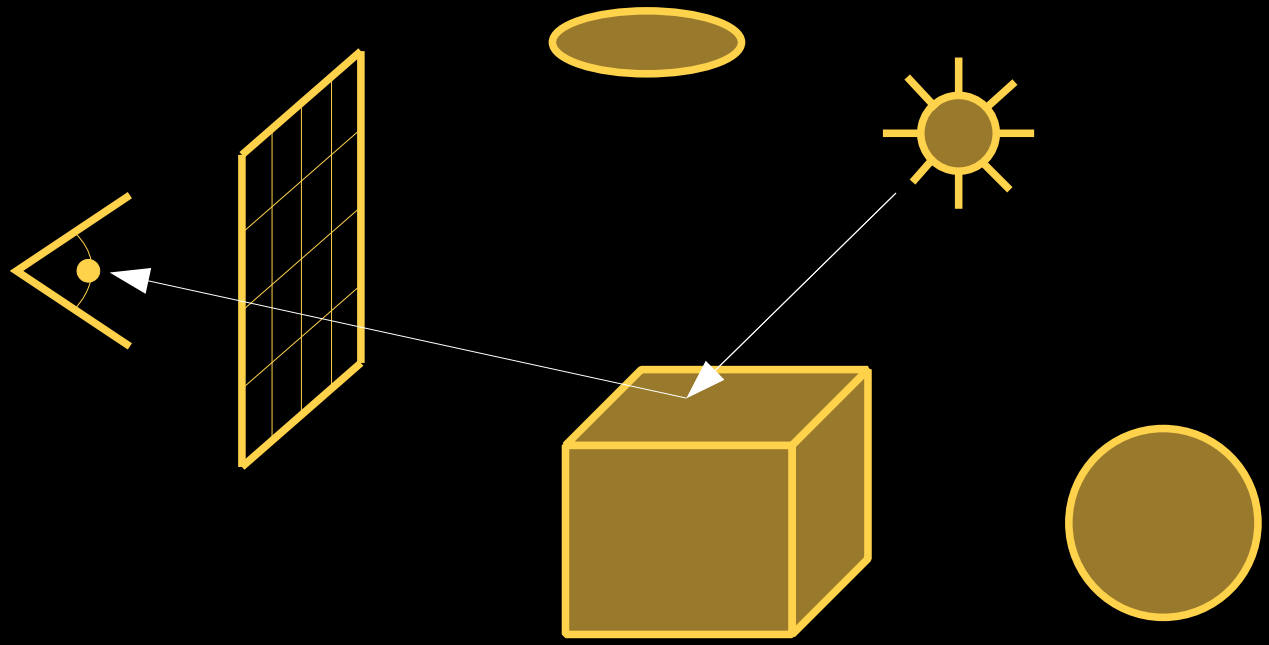
- Problème : la plupart des rayons n'atteignent pas l'oeil
  - Calculs inutiles : méthode très inéfficace !
  - Comment choisir les rayons pour atteindre uniformément tous les pixels de l'écran ?



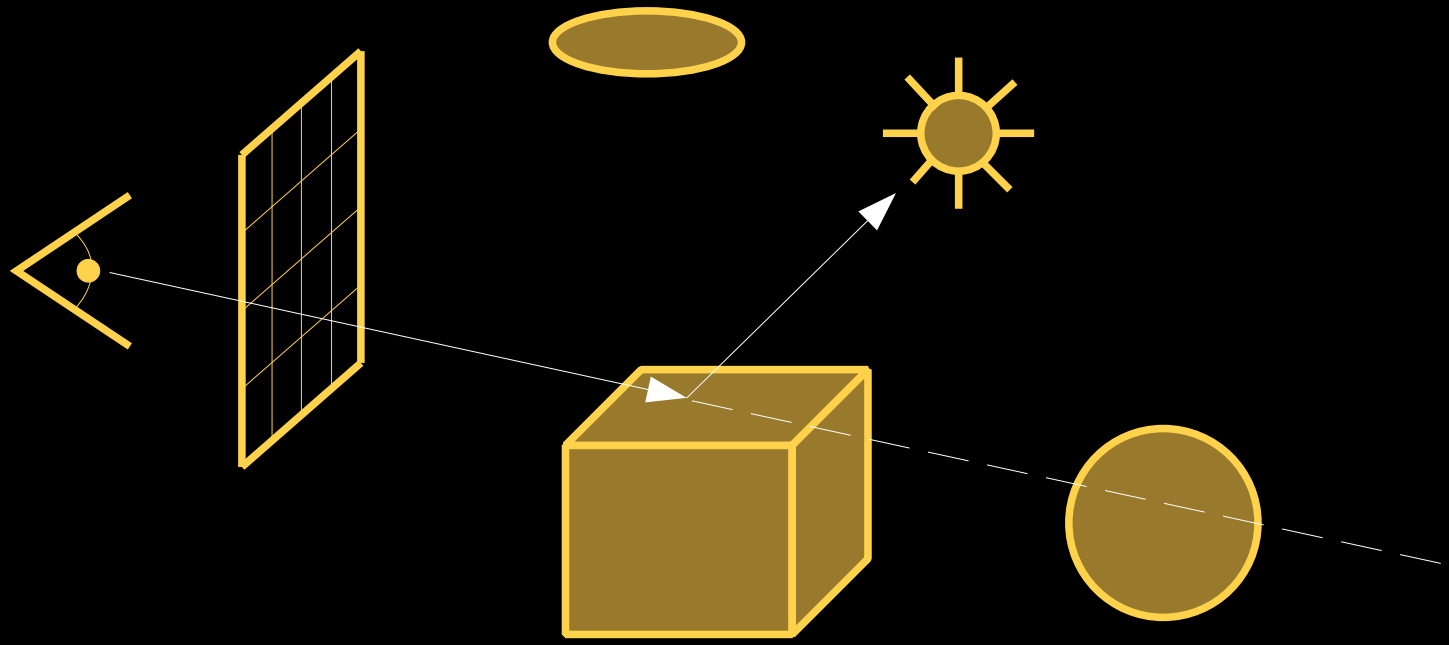
Algorithme du lancer de rayon

- Principe du retour inverse de la lumière :
  - si la lumière suit un trajet quelconque d'un point A à un point B, alors la lumière peut suivre exactement le trajet inverse de B vers A
- Idée
  - Inverser le sens des rayons : partir de l'oeil
- Avantage :
  - On ne calcule que ce qui est visible !
  - Tous les pixels sont couverts uniformément

Algorithme du lancer de rayon

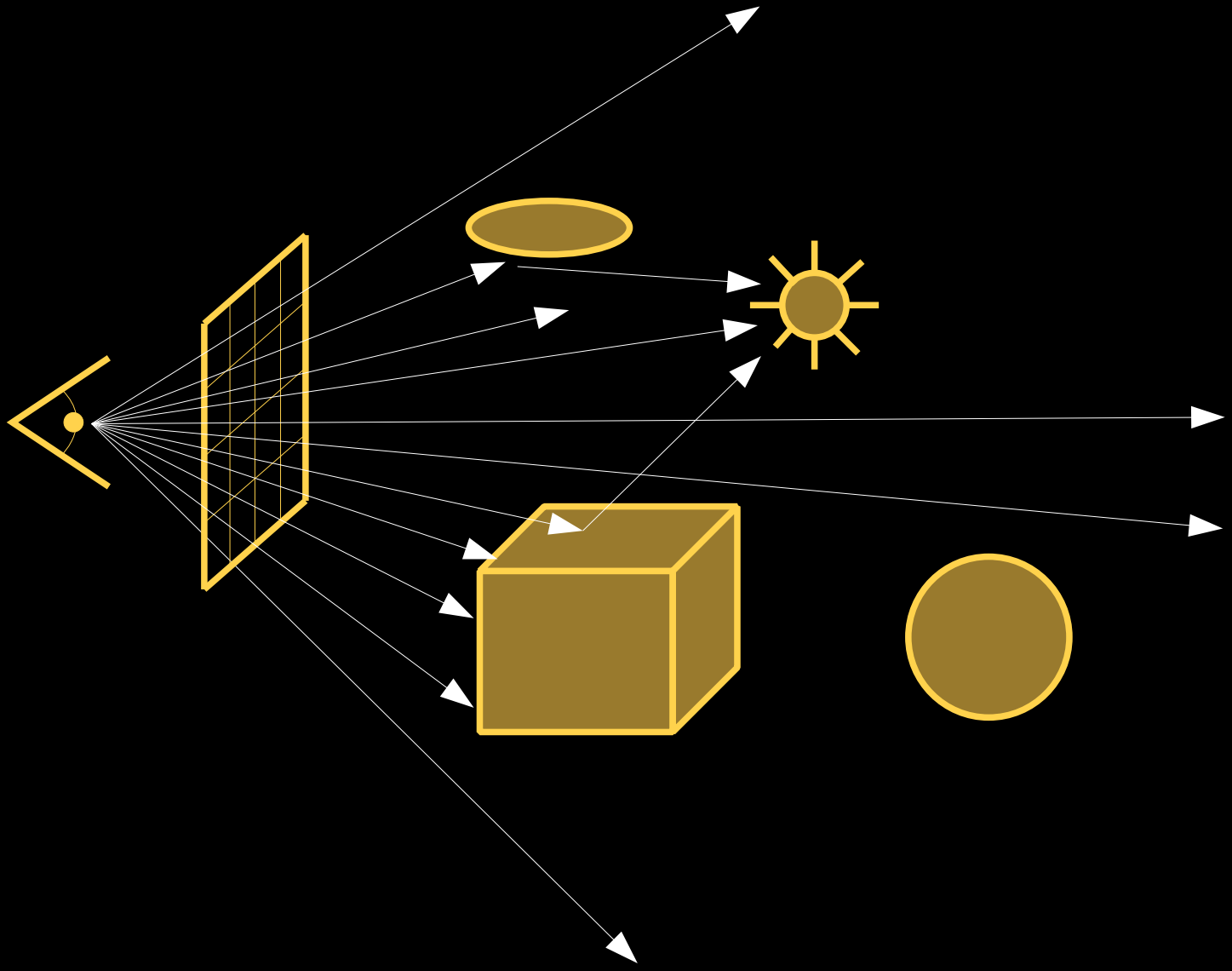


Algorithme du lancer de rayon



On garde l'intersection la plus proche de l'oeil.

Algorithme du lancer de rayon



Algorithme du lancer de rayon

- Rayons bien répartis sur les pixels
  - Exemple : un rayon par pixel
- On calcule quel objet s'affiche en chaque pixel
  - Alternative à la rasterization

Lancer de rayon Vs Rasterization

- Rasterization:
  - Pour tous les triangles :
    - trouver les rayons qui intersectent le triangle
- Raytracing:
  - Pour tous les rayons :
    - trouver les triangles qui intersectent le rayon

Lancer de rayons : pseudo-code

- Pour chaque rayon partant de l'oeil
  - Zmin = infini
  - Pour chaque primitive
    - Si rayon intersecte primitive en P
      - Si (zp < zmin) alors I = P
  - Couleur rayon = couleur en I

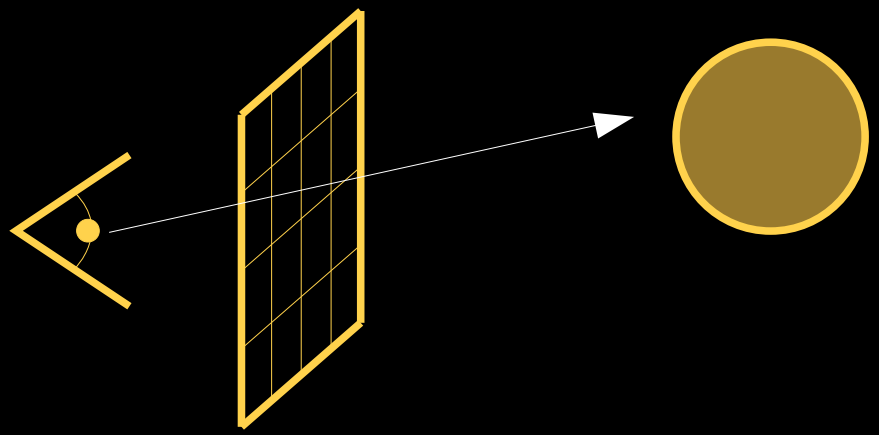
- Parallèlisme ?
- Complexité ?

Lancer de rayons : pseudo-code

- Pour chaque rayon partant de l'oeil
  - Zmin = infini
  - Pour chaque primitive
    - Si rayon intersecte primitive en P
      - Si (zp < zmin) alors I = P
  - Couleur rayon = couleur en I

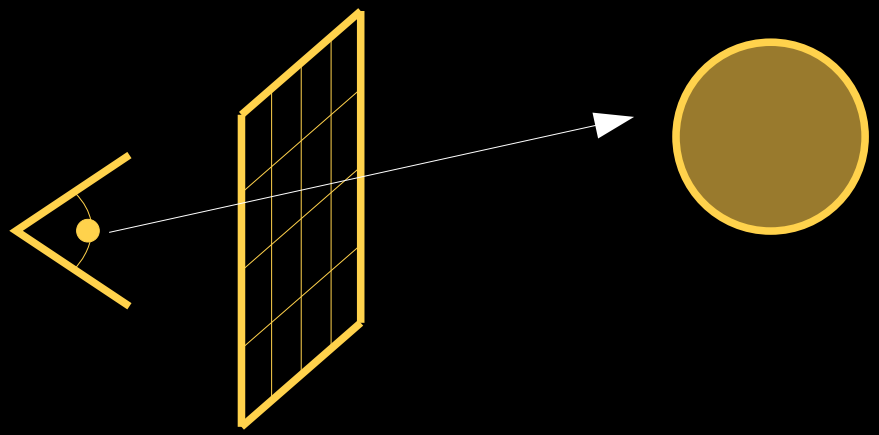
- Parallèlisme : chaque rayon est indépendant
- Complexité : *O(Nb rayons * Nb primitives)*
  - En pratique : 10^6 * 10^6 … on verra comment accélérer

Exercice



- Soit :
  - un rayon défini par son origine O et sa direction **d**
  - une sphère de rayon R et de centre **C**
- Déterminer la position de l'intersection entre le rayon et la sphère, si elle existe.
- Comment faire quand il y a deux intersections ?

Exercice (solution)



- Soit $\mathbf{P}$ un point d'intersection
- $\mathbf{P}$ est sur le rayon : $\mathbf{P} = \mathbf{O} + t\mathbf{d}$ avec $t>0$
- $\mathbf{P}$ est sur la sphère : $|| \mathbf{P} - \mathbf{C} ||^2 = R^2$
- $\mathbf{d}^2t^2 - 2\,\mathbf{OC}.\mathbf{d}\,t + \mathbf{OC}^2 - R^2 = 0$
  - La plus petite racine positive (si elle existe) donne l'intersection recherchée

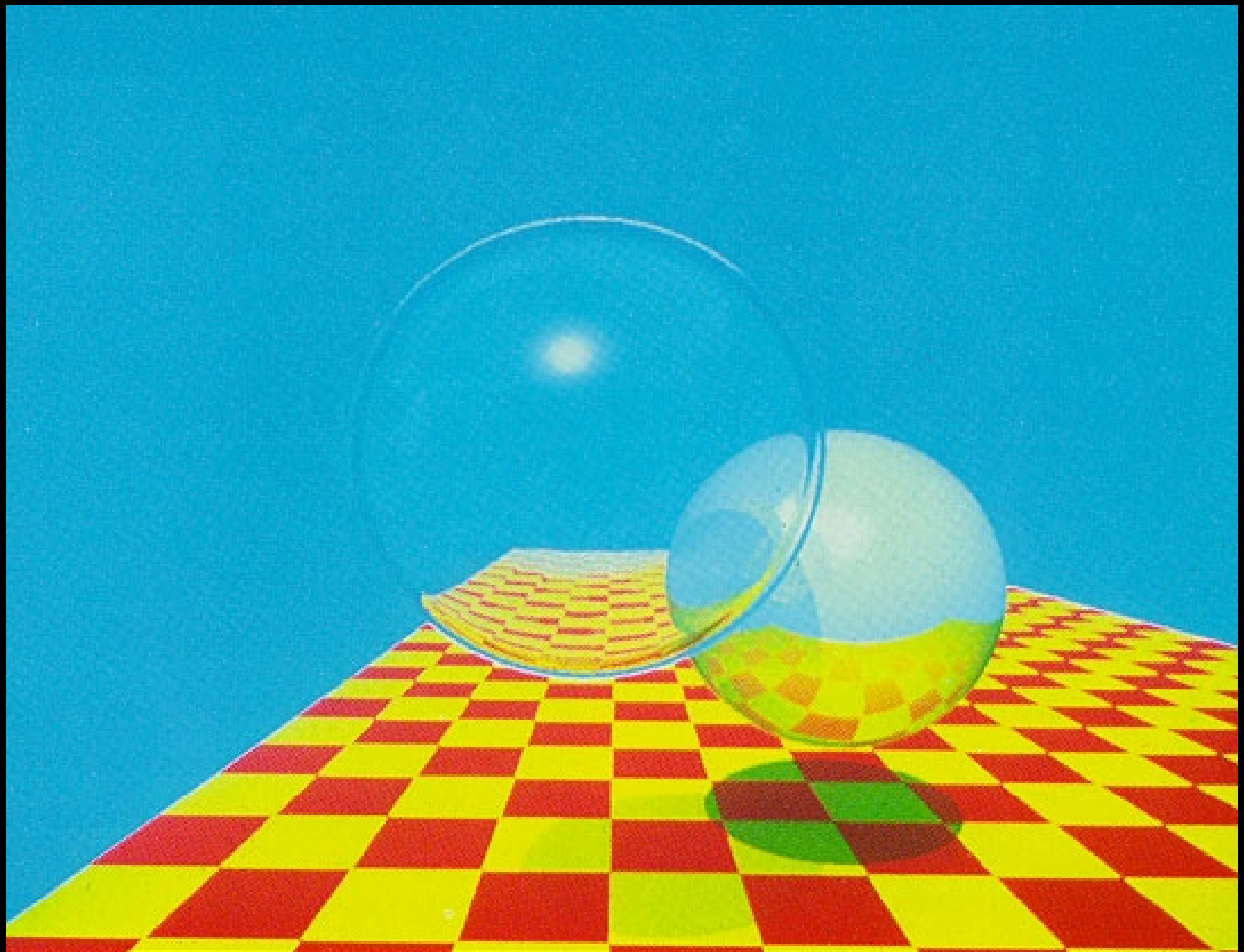
Primitives géométriques

- Le lancer de rayon peut rendre
  - Formes géométriques simples : sphères, cylindres ...
  - Polygones (triangles ...)
  - Surfaces de subdivision, splines ...
  - CSG
  - Données volumiques
  - N'importe quel objet que vous savez intersecter avec un rayon !

Calcul de l'éclairage

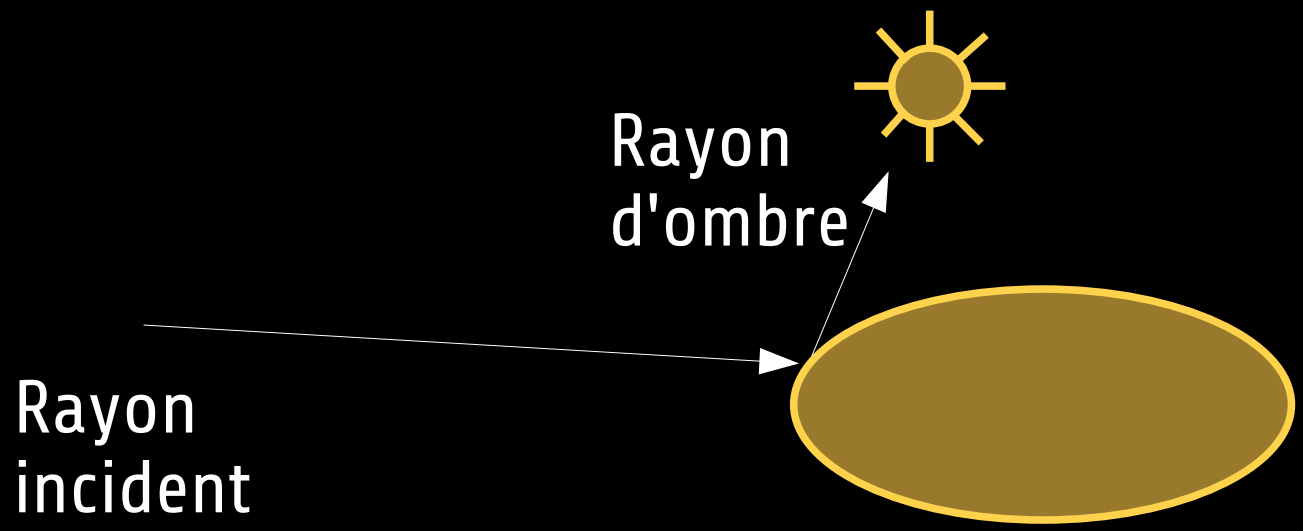
- Modèle simple (Phong)
- Modèle amélioré : Whitted
  - *An improved Illumination Model for Shaded Display*, T.Whitted, 1980.
    - Ombres
    - Reflets spéculaires
    - Réfractions spéculaires (transparence)

Modèle de Whitted



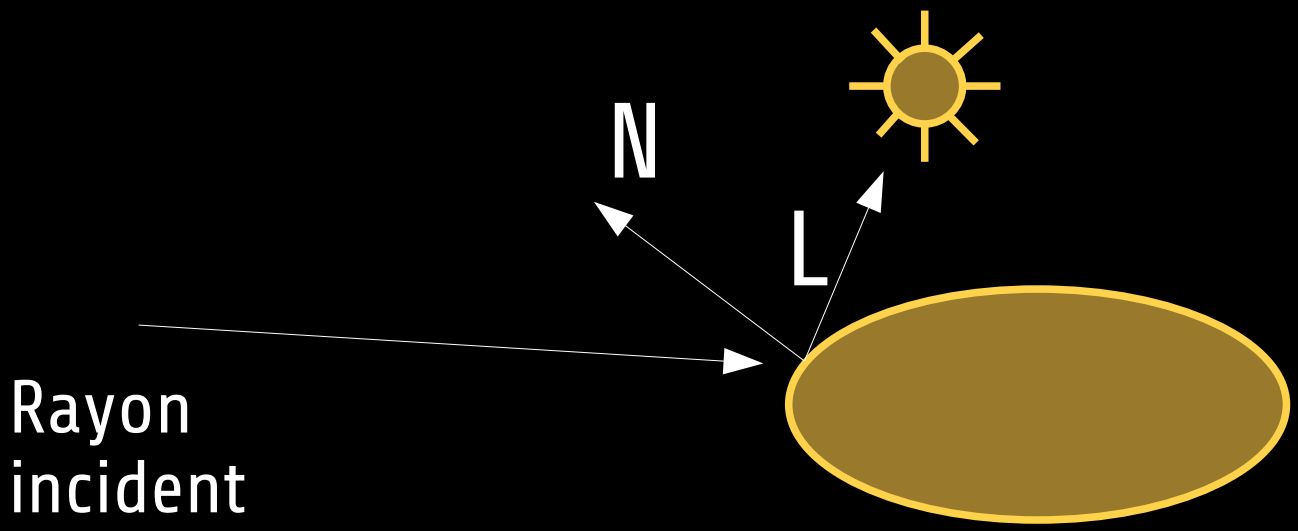
Modèle de Whitted

- Cas Rayon – Surface diffuse
  - On lance un rayon d'ombre vers la source



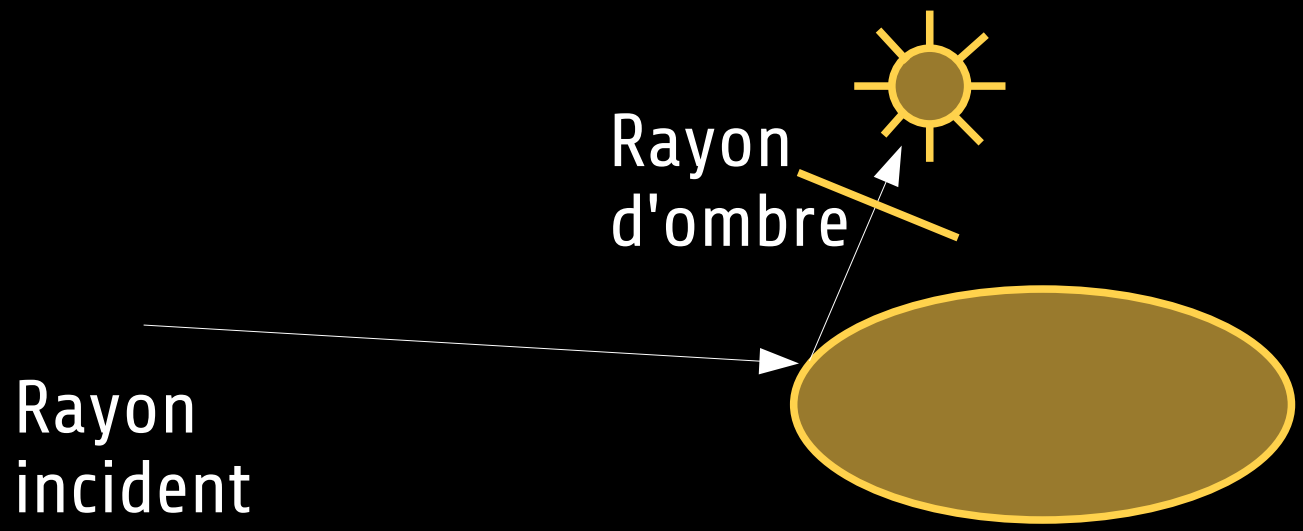
Modèle de Whitted

- Cas Rayon – Surface diffuse
  - On lance un rayon d'ombre vers la source
  - Si pas d'intersection : on se ramène au modèle de Phong
    - Calcul de la couleur en utilisant N et L



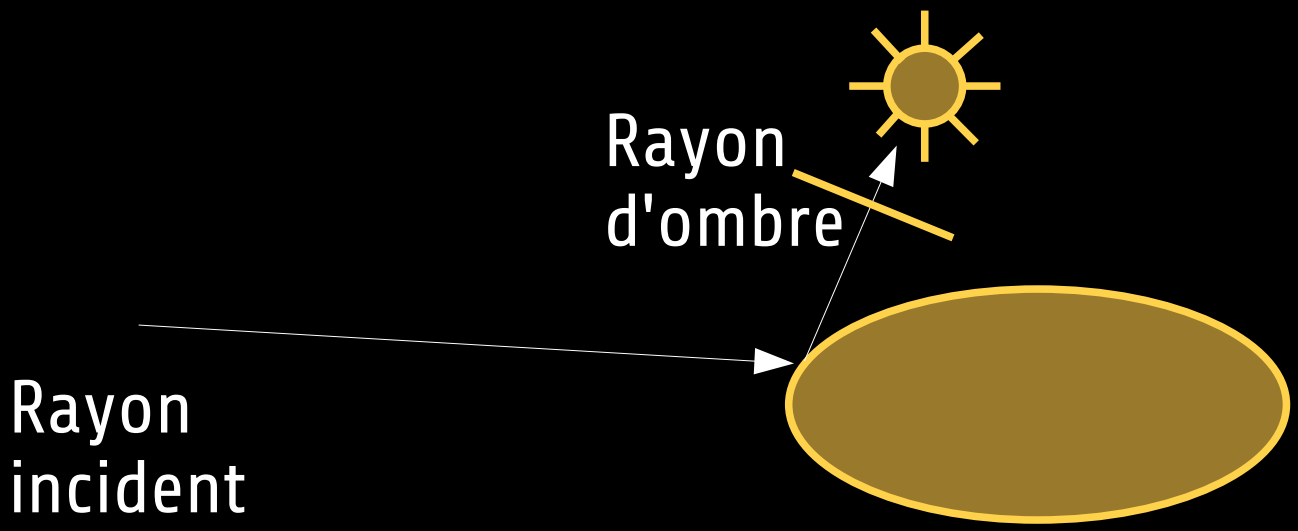
Modèle de Whitted

- Cas Rayon – Surface diffuse
  - On lance un rayon d'ombre vers la source
  - Si pas d'intersection : on se ramène au modèle de Phong
  - Si intersection : à l'ombre



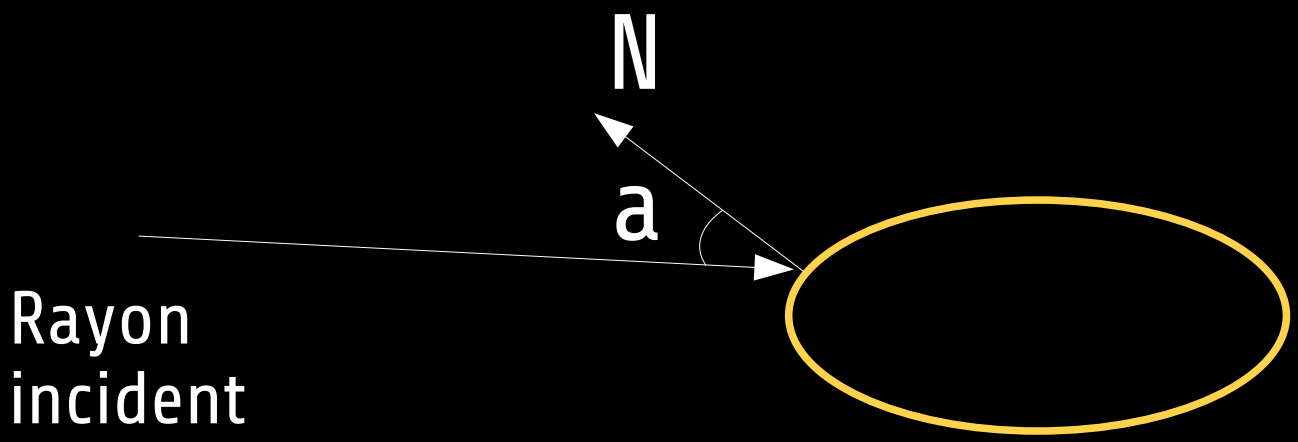
Rayon d'ombre

- Test booléen d'intersection
  - Pas besoin de calculer les coordonnées d'intersection
- Attention aux erreurs de précision
  - Chercher une intersection avec $t > \varepsilon$



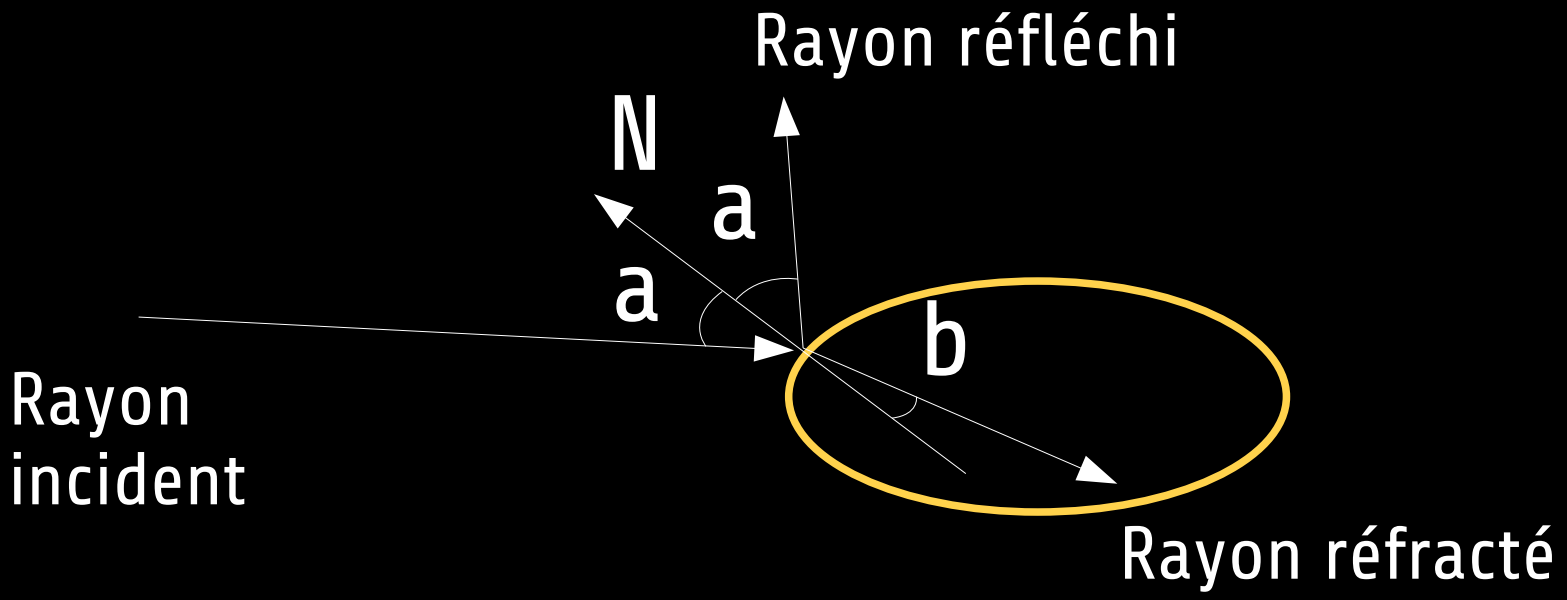
Modèle de Whitted

- Cas Rayon – Surface spéculaire parfaite
  - Exemples : mirroir, verre …
  - Avez vous une idée ?



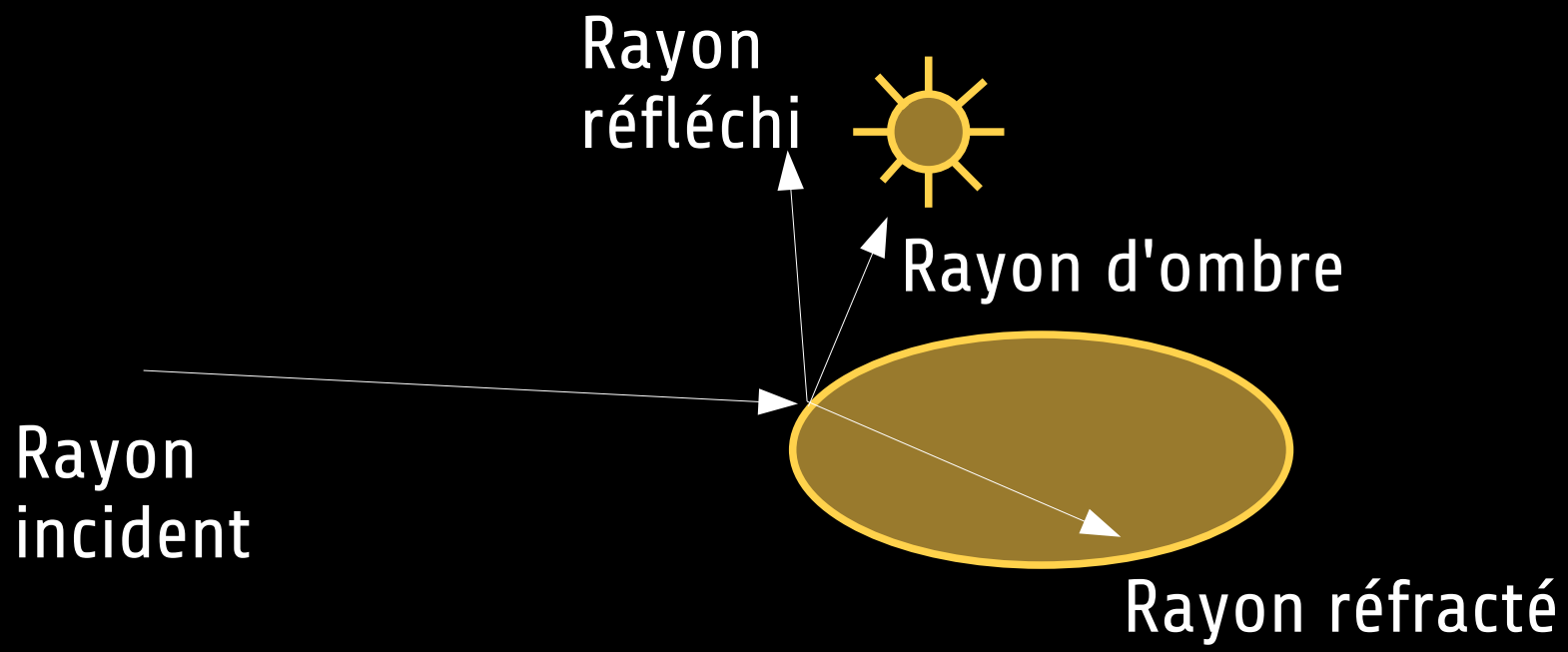
Modèle de Whitted

- Cas Rayon – Surface spéculaire parfaite
  - On génère un rayon réfléchi et un rayon réfracté
  - Loi de Snell-Descartes : n sin(a) = m sin(b)
  - On lance récursivement ces rayons dans la scène



Modèle de Whitted

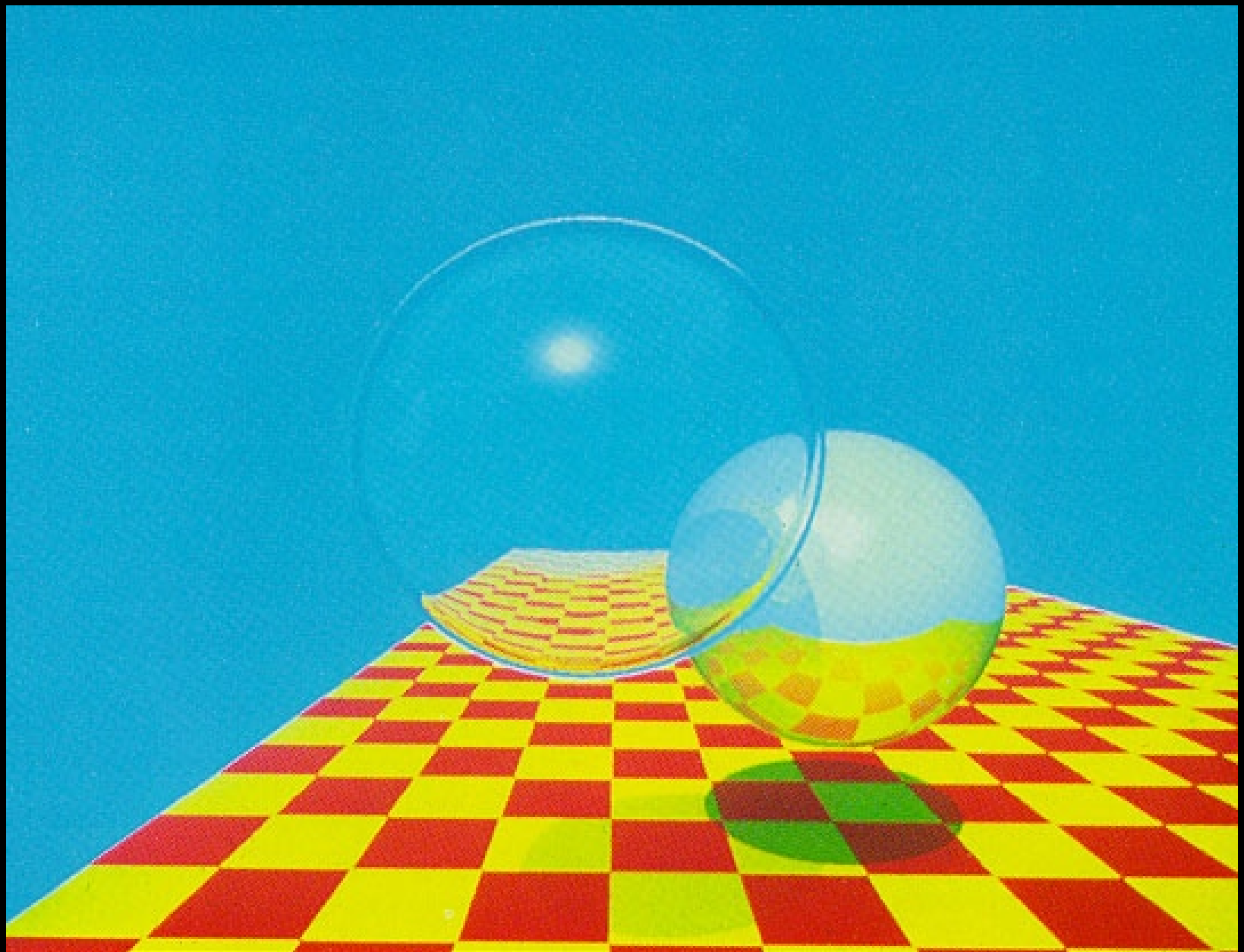
- Cas Rayon – Surface mixte
  - Cas le plus général : 3 rayons générés



Modèle de Whitted

- Chaque rayon peut engendrer 3 rayons
  - Arbre de rayons
- Quand est-ce que ça s'arête ? 2 options :
  - Après un nombre fixe de rebonds
  - Ou lorsque le rayon transporte une énergie < epsilon

Modèle de Whitted



Modèle de Whitted

Limites du modèle de Whitted

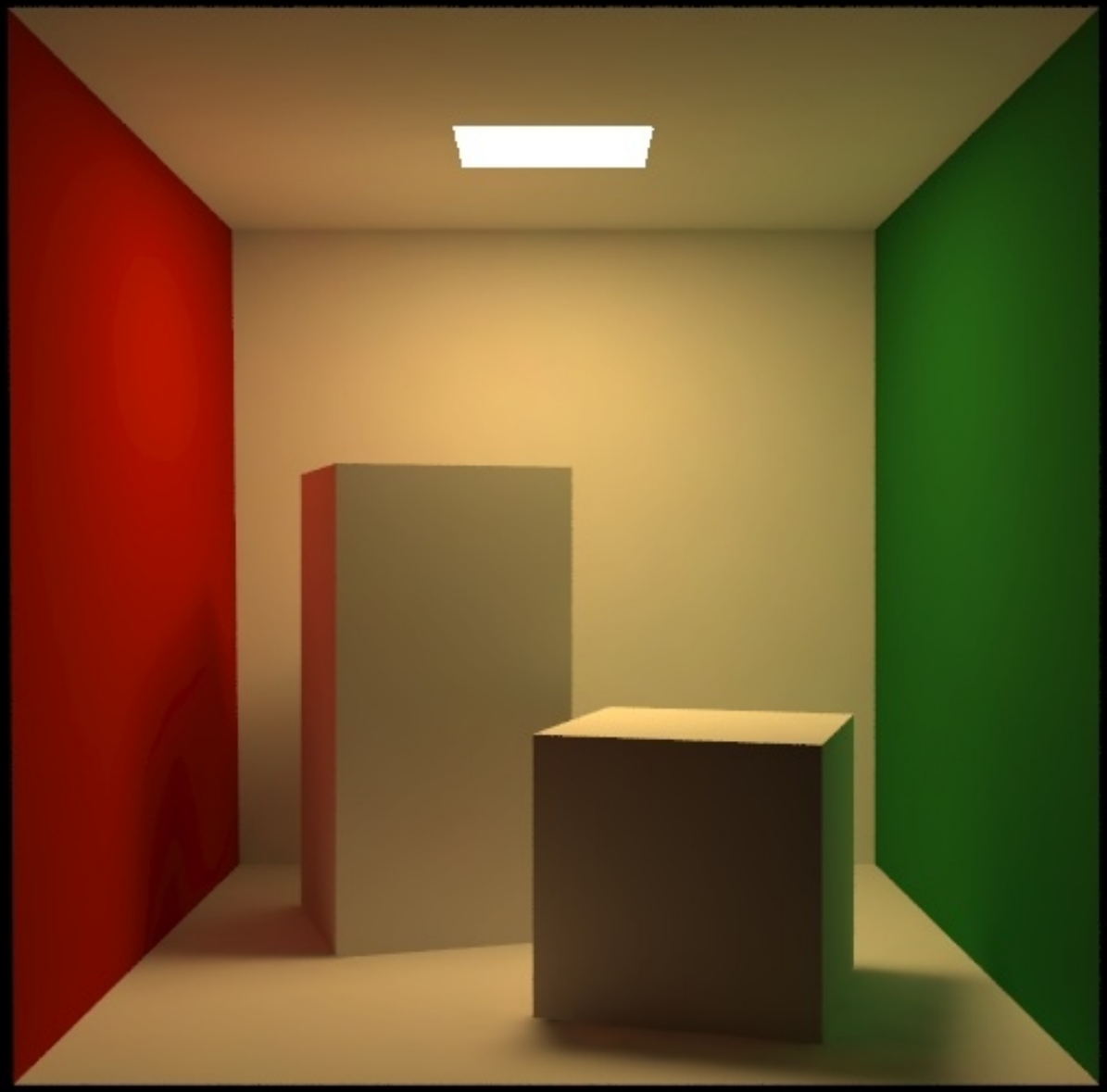
- Le modèle de Whitted peut représenter les chemins lumineux du type [O S* D L]
  - O : Oeil
  - S : surface Spéculaire
  - D : surface Diffuse
  - L : source Lumineuse
- A quels effets correspondent les chemins :
  - [O D* L] ?
  - [O D S* L] ?

Limites du modèle de Whitted

Caustics

Limites du modèle de Whitted

Color Bleeding



Limites du modèle de Whitted

- Le modèle de Whitted peut représenter les chemins lumineux du type [O S* D L]
  - O : Oeil
  - S : surface Spéculaire
  - D : surface Diffuse
  - L : source Lumineuse
- A quels effets correspondent les chemins :
  - [O D* L] ? *Color Bleeding*
  - [O D S* L] ? *Caustiques*
- Comment obtenir ces effets ?

Résumé

- Large choix de primitives
- Calcule uniquement ce qui est affiché
- Ombres et effets spéculaires
  - Lancer de rayons récursif
- Forte complexité algorithmique pour la version de base
  - O(nb rayons * nb primitives)
  - On verra comment aller plus vite
- Algorithme naturellement parallèle

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

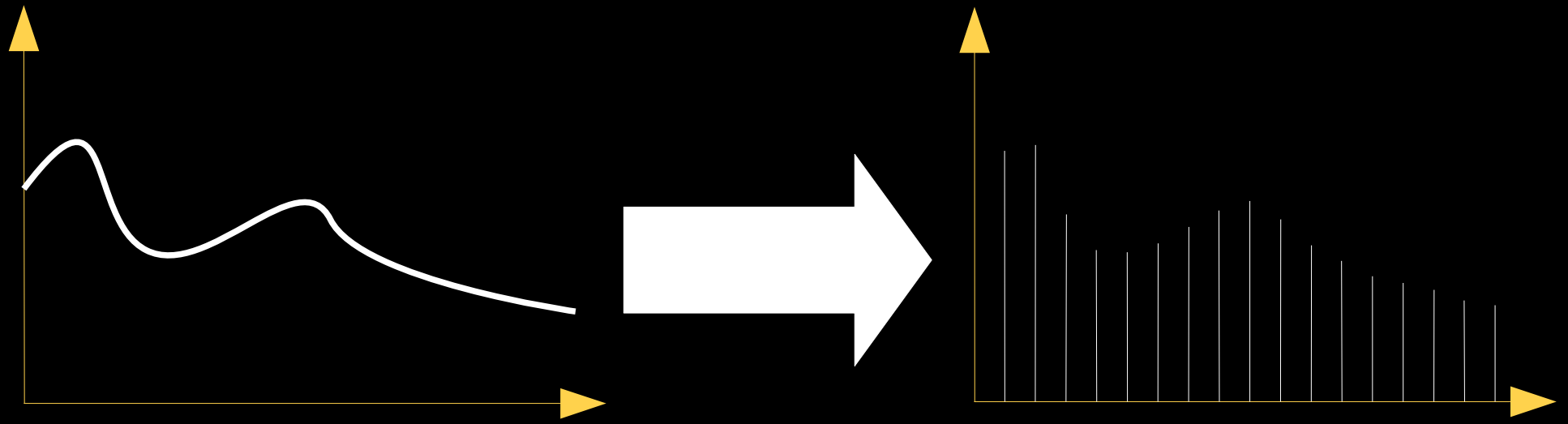
Aliasing

- Problèmes liés à l'échantillonnage
  - Echantillonnage = approximer une fonction par un nombre fini d'échantillons
  - Plusieurs fonctions peuvent avoir le même échantillonnage : ambigüité
- Indépendant du lancer de rayons
  - On reviendra au lancer de rayons plus tard

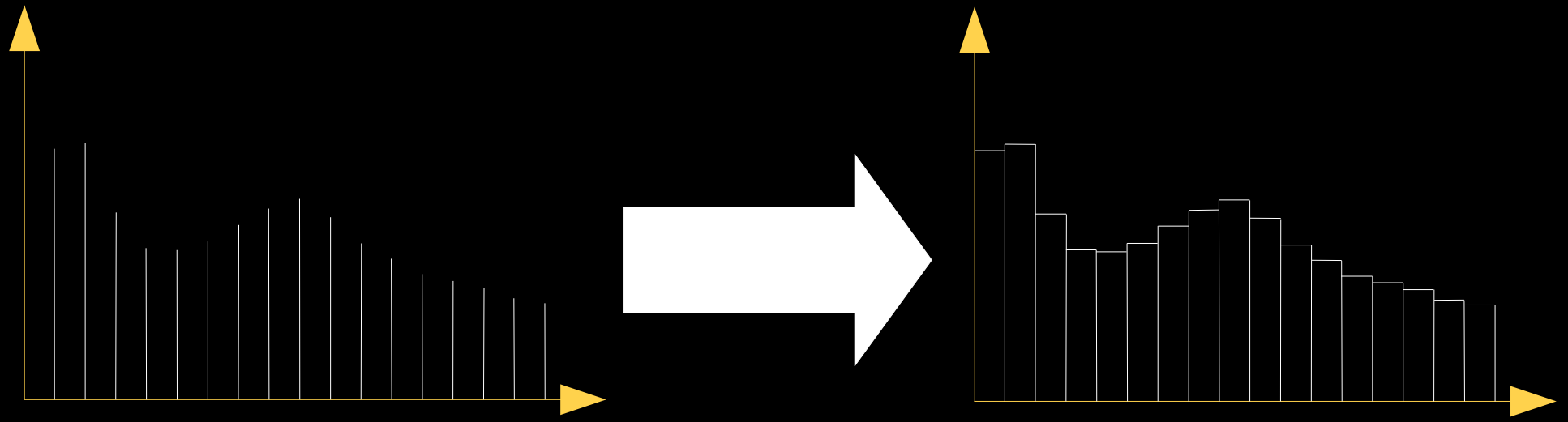
Reconstruction

- Reconstituer le signal d'origine à partir des échantillons
- Utilité en graphisme :
  - Zoom sur une image

Echantillonnage

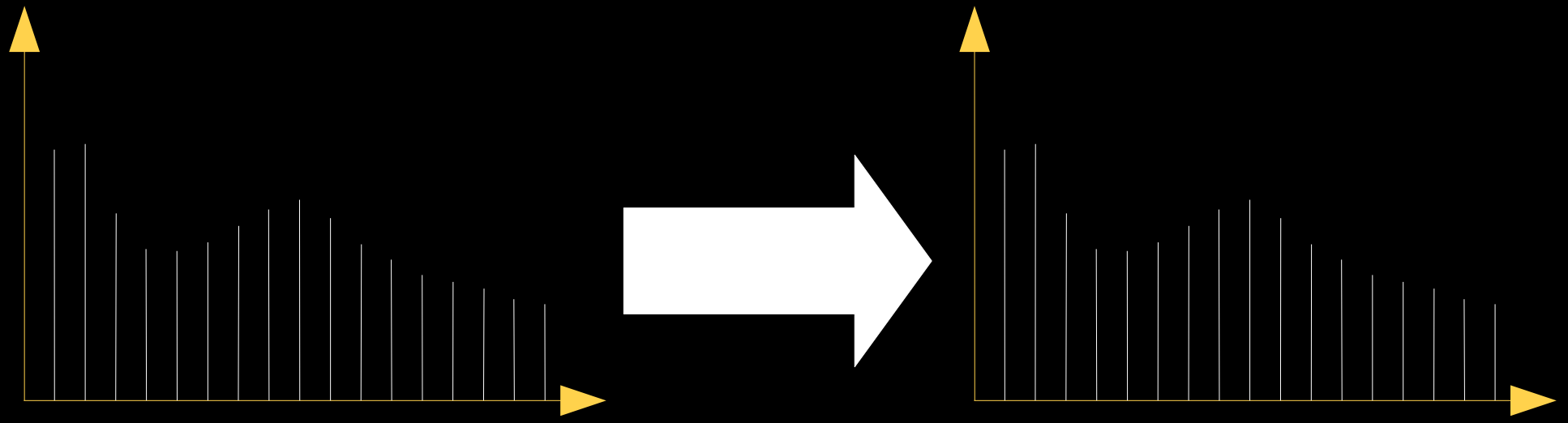


Reconstruction : créneau



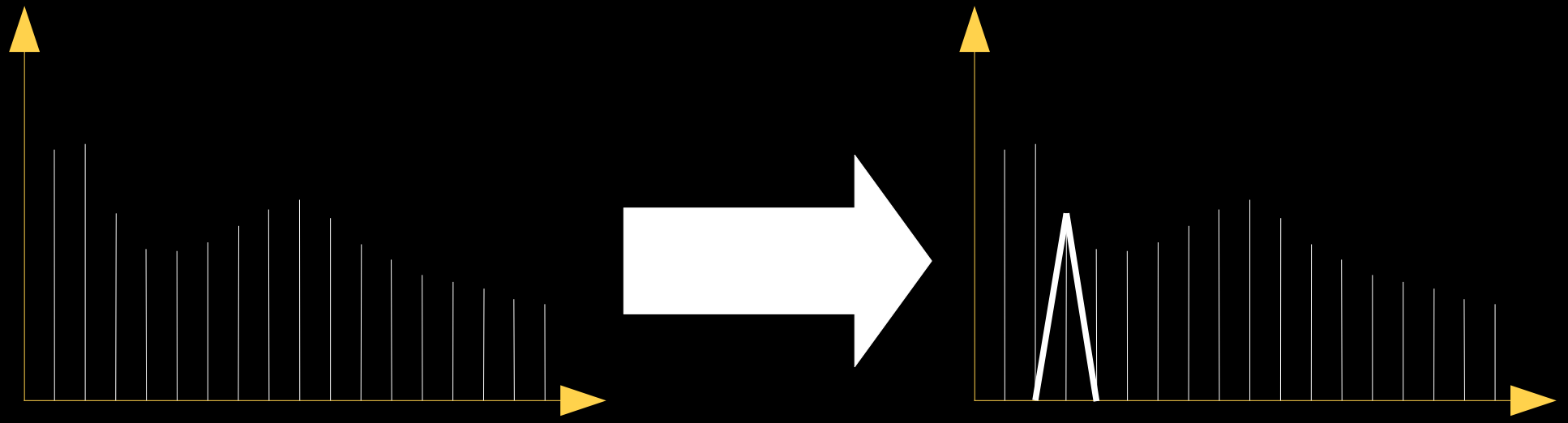
Filtre créneau : chaque échantillon est remplacé par un créneau. Le signal reconstruit est la somme des créneaux.

Reconstruction : triangle



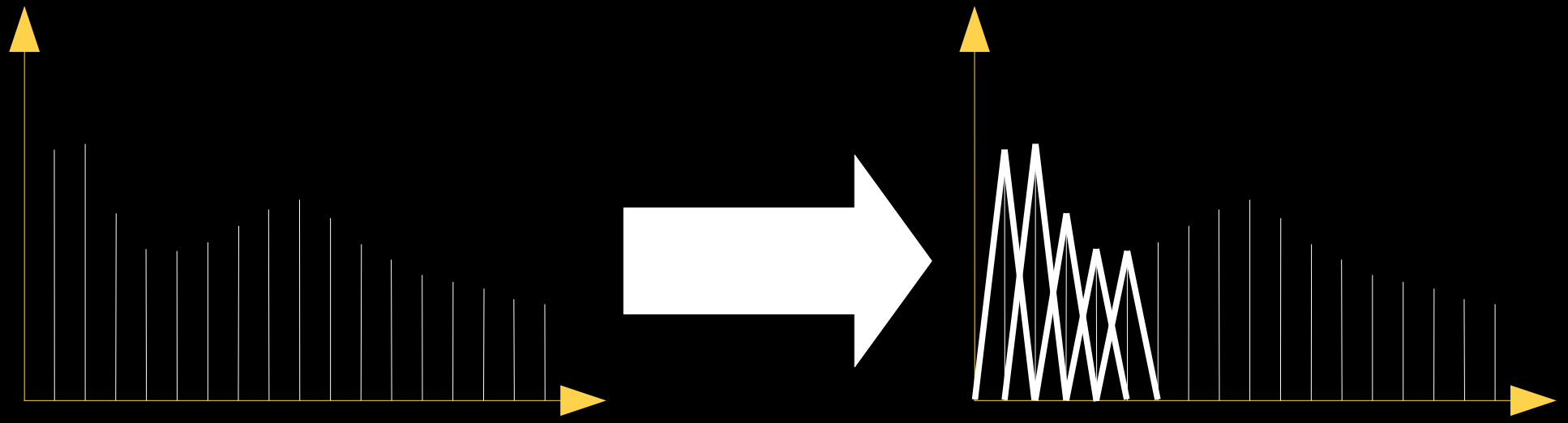
Filtre triangle : chaque échantillon est remplacé par un triangle. Le signal reconstruit est la somme des triangles.

Reconstruction : triangle



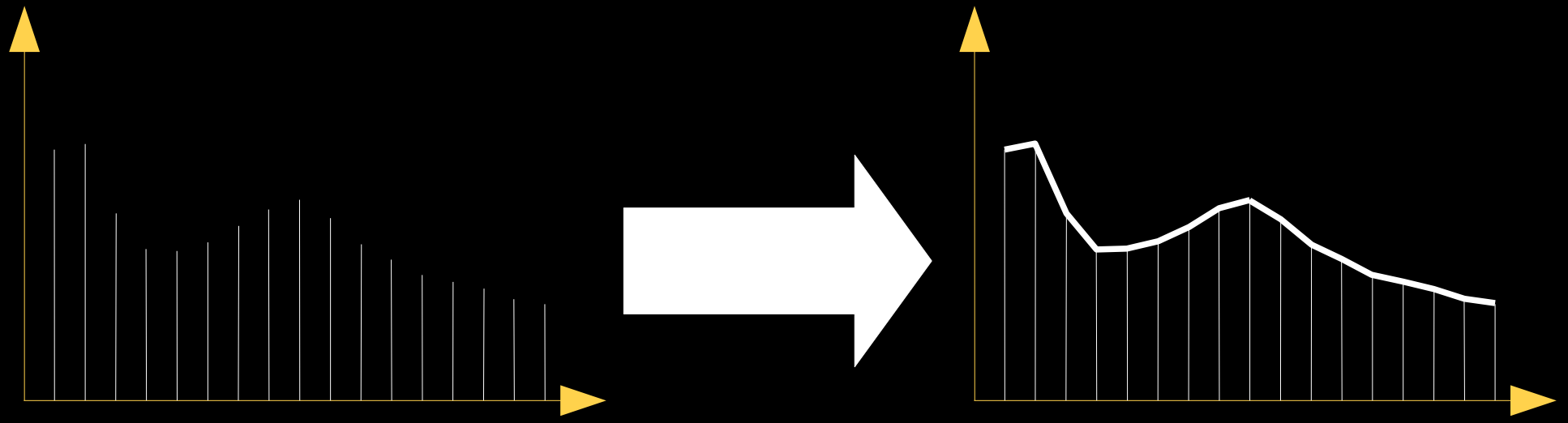
Filtre triangle : chaque échantillon est remplacé par un triangle. Le signal reconstruit est la somme des triangles.

Reconstruction : triangle



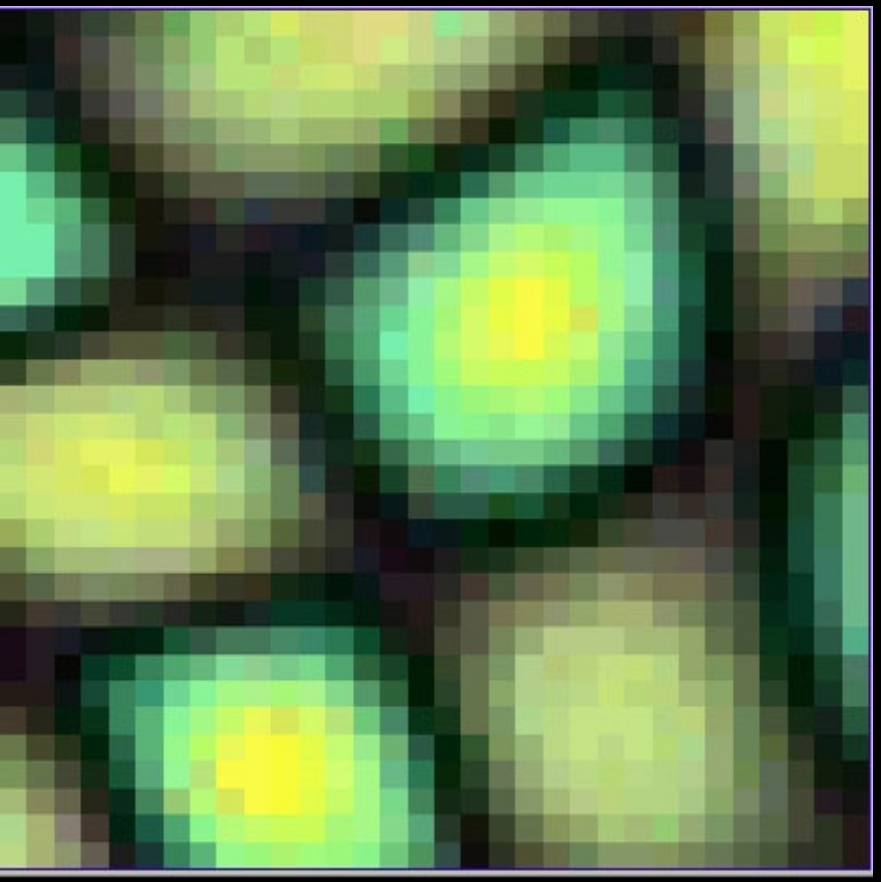
Filtre triangle : chaque échantillon est remplacé par un triangle. Le signal reconstruit est la somme des triangles.

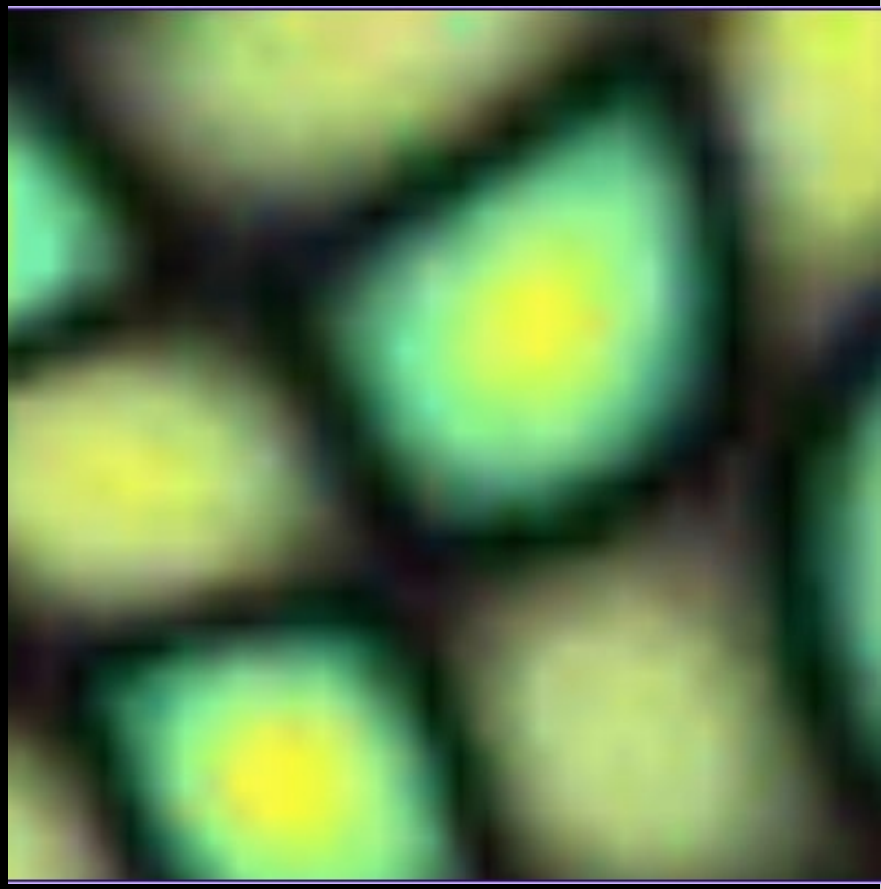
Reconstruction : triangle



Filtre triangle : chaque échantillon est remplacé par un triangle. Le signal reconstruit est la somme des triangles. On obtient une fonction linéaire par morceau.

Reconstruction





A gauche : créneau (GL_NEAREST)
A droite : triangle (GL_LINEAR)

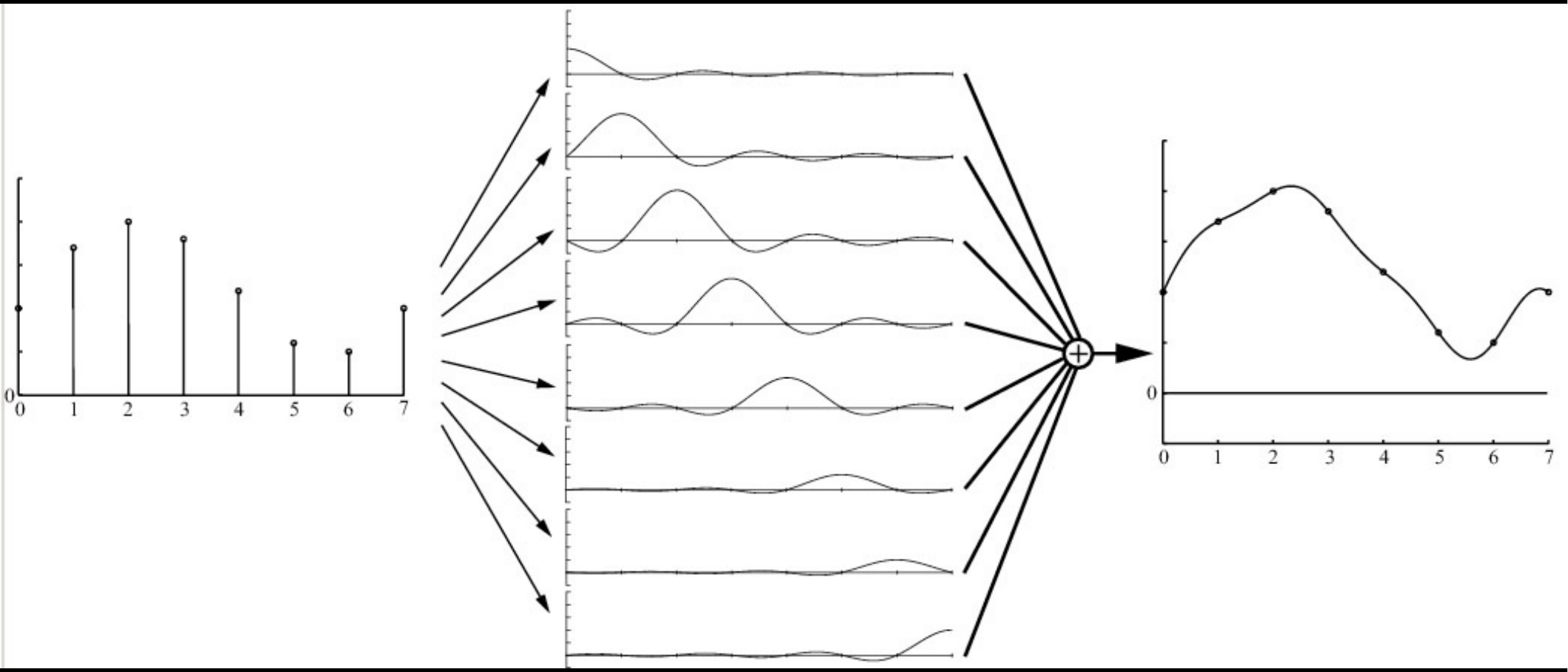
Reconstruction : sinc

- Théorème de Nyquist :
  - La fréquence d'échantillonnage doit être au moins le double de la plus grande fréquence présente dans le signal
- La fréquence d'échantillonnage limitée
- Fréquence de l'image infinie (arêtes ...)
- L'oeil humain perçoit surtout les basses fréquences

Reconstruction : sinc

- Mode d'emploi
  - Prendre plus d'échantillons (plusieurs par pixel)
  - Supprimer les hautes fréquences
    - Utiliser le filtre de reconstruction sinc (passe-bas)
    - sinc(x) = sin(x) / x
    - Ca a tendance à rendre l'image floue, mais on ne peut pas faire mieux

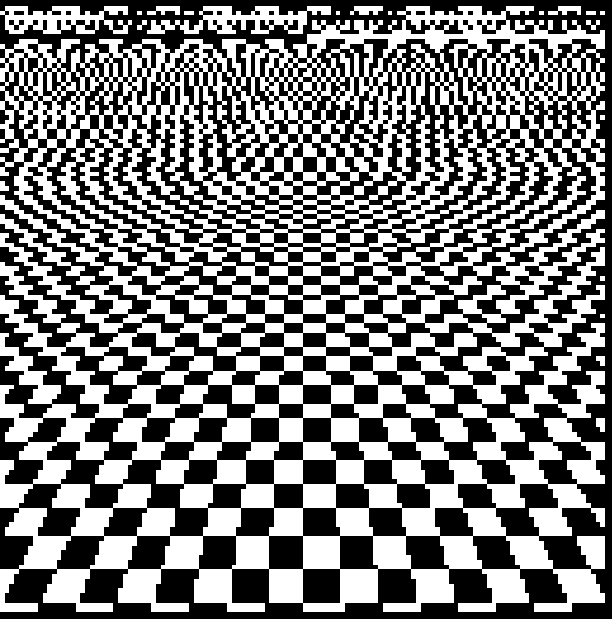
Reconstruction : sinc



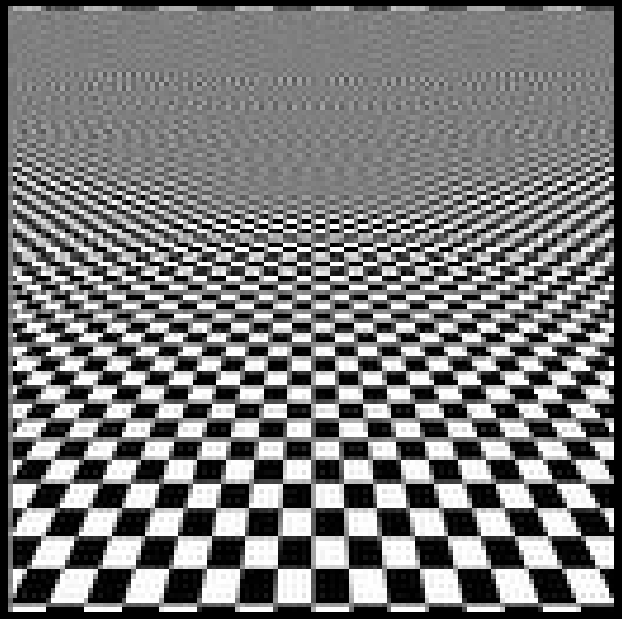
Reconstruction : sinc

- Difficile :
  - Support infini
  - Valeurs négatives
- En pratique :
  - Approximations
  - Gaussienne
  - Filtre cubique

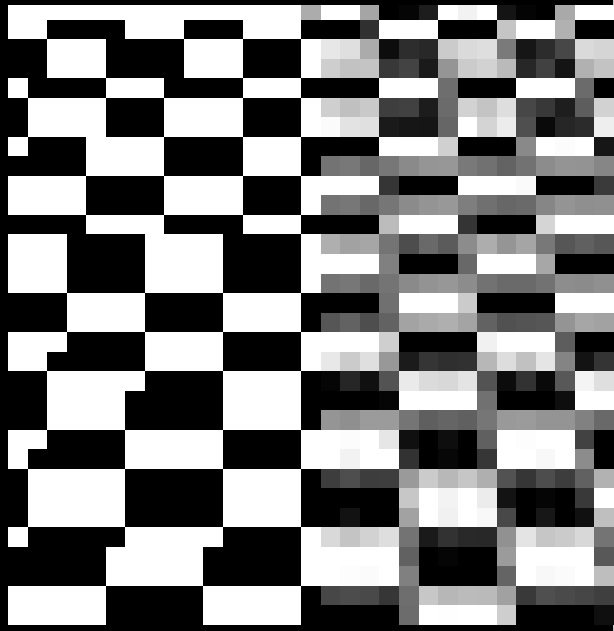
Exemples d'échantillonnages



Un seul échantillon au centre des pixel



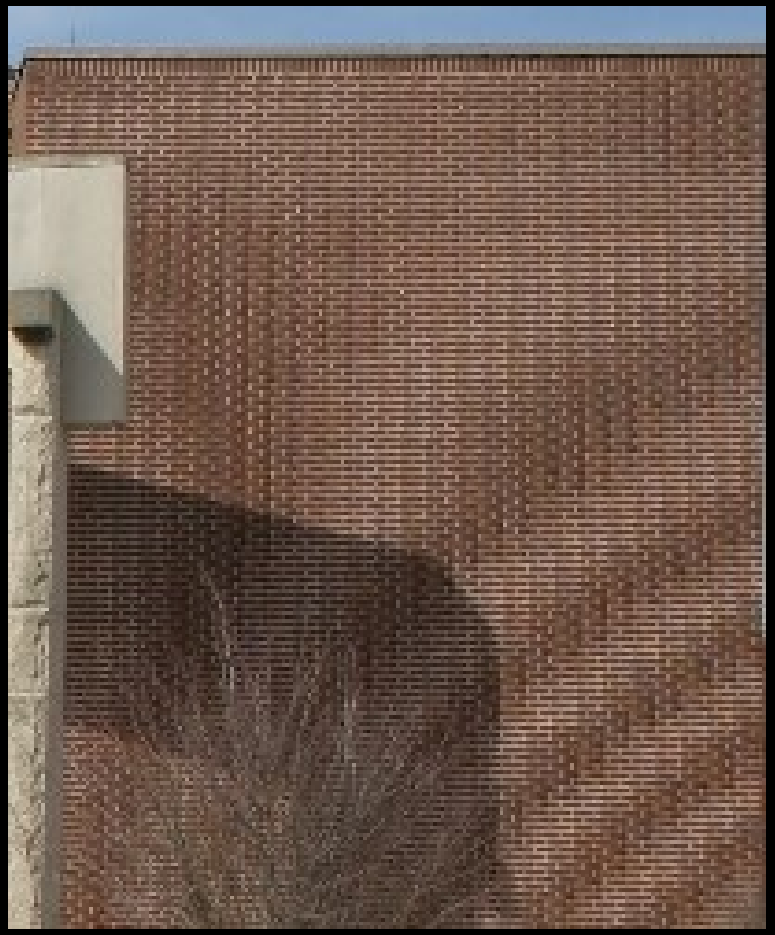
Plusieurs échantillons par pixel puis reconstruction par filtre sinc



Comparaison (Zoom)

Exemples d'échantillonnages

Plusieurs échantillons par pixel



Un seul échantillon par pixel : effet moiré

Retour au lancer de rayons

- En pratique :
  - On lance plusieurs rayons par pixel
  - On fait la moyenne (pondérée éventuellement)
- Comment choisir les positions des rayons dans chaque pixel ?

Supersampling

Grille régulière :
Oeil humain sensible aux motifs réguliers : artefacts possibles

Aléatoire :
On remplace le moiré par du bruit
Pas de garantie sur la répartition des points

Supersampling

Poisson disc : aléatoire avec garantie d'espacement
Trop coûteux en pratique

Jittering :
aléatoire à l'intérieur d'une grille
Efficace en pratique

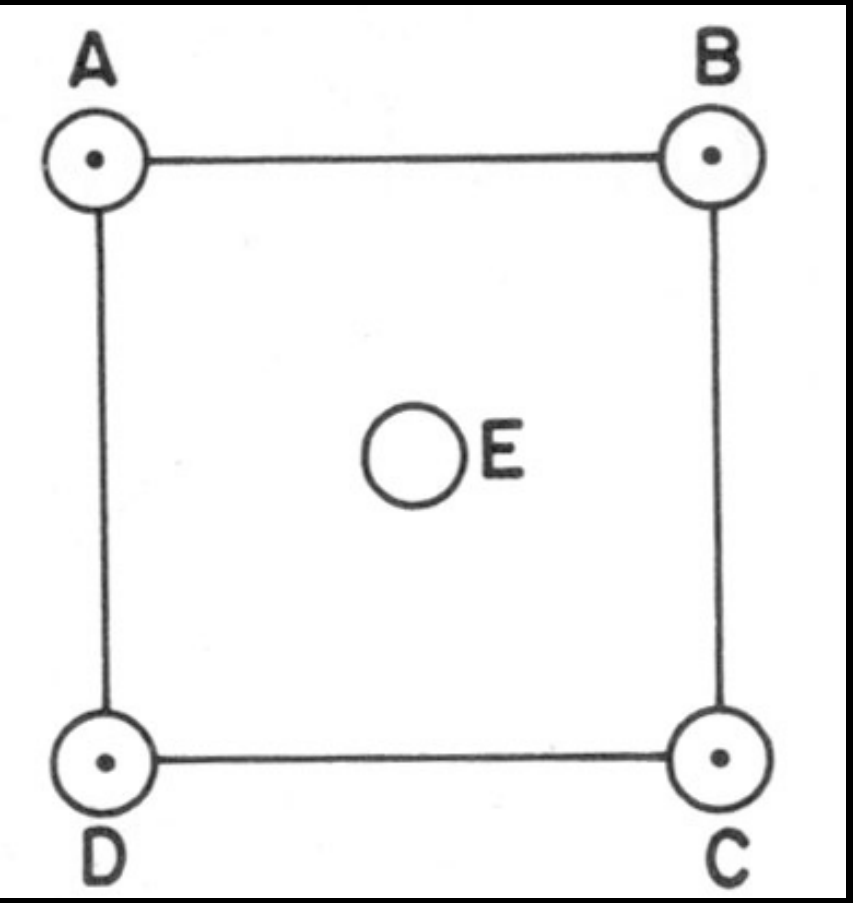
Supersampling

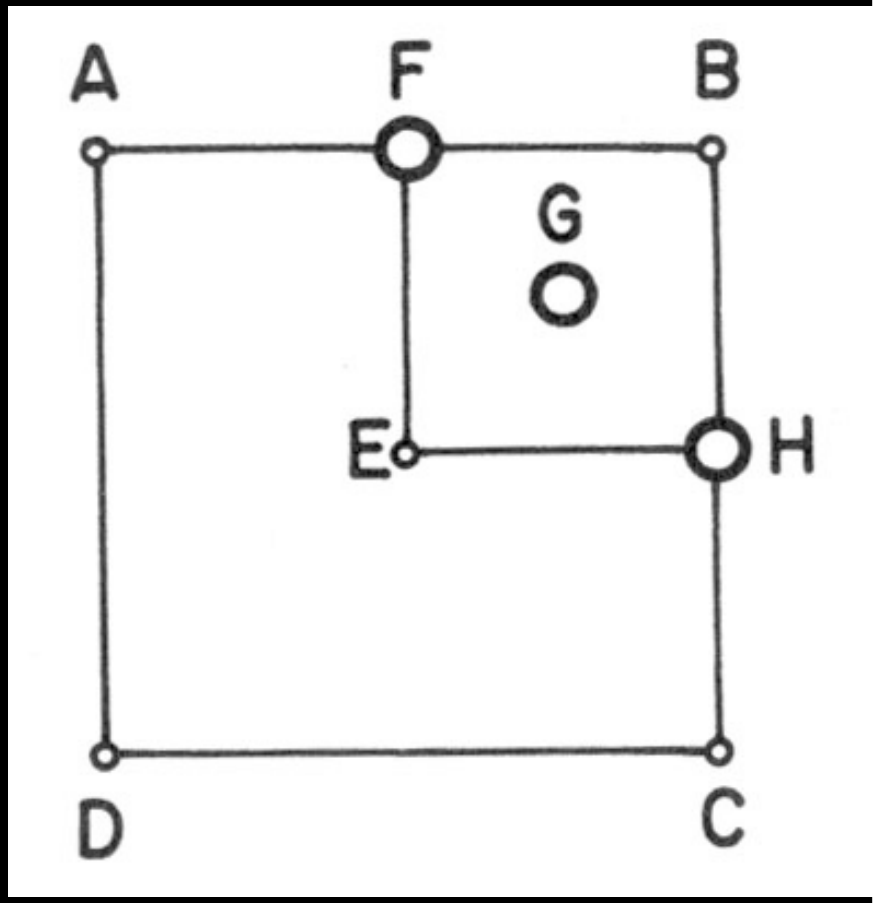
Grille tournée :
mieux que grille alignée sur les axes

Supersampling Adaptatif

- Nyquist :
  - Les zones de hautes fréquences ont besoin de beaucoup de rayons
  - Les zones de basses fréquences ont besoin de peu de rayons
- Whitted : méthode de supersampling adaptatif pour le lancer de rayons

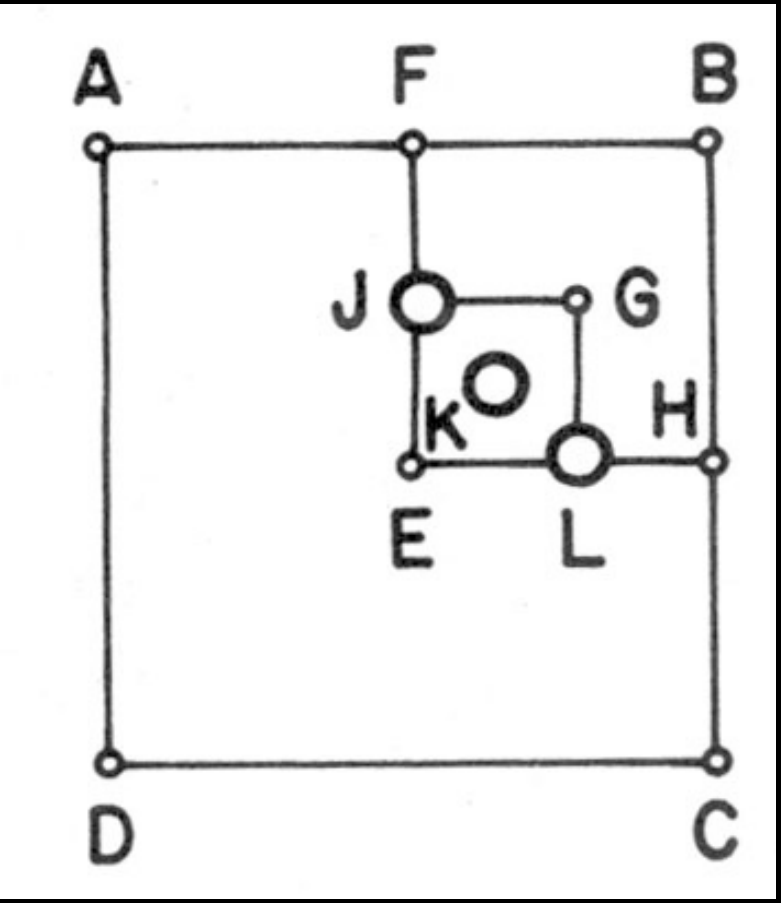
Supersampling Adaptatif





Lancer 5 rayons : A,B,C,D et E
Comparer les couleurs obtenues.
Si B et E ont une couleur trop differente, lancer F, G et H.

Supersampling Adaptatif



Appliquer récursivement, jusqu'à ce que tout couple de rayons adjacents aient une couleur proche,
ou jusqu'à une profondeur limite.

Donner aux rayons un poids proportionnel à leur surface couverte.

Supersampling Adaptatif

- Très simple en lancer de rayons
- Impossible à faire en rasterization
- Efficace en pratique
- Ca reste du super sampling
  - Nyquist

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Rappel

- Pour chaque rayon partant de l'oeil
  - Zmin = infini
  - Pour chaque primitive
    - Si rayon intersecte primitive en P
      - Si (zp < zmin) alors I = P
  - Couleur rayon = couleur en I
  - Générer rayon d'ombre et rayons spéculaires récursivement

- Complexité : *O(Nb rayons * Nb primitives)*
  - En pratique : 10^6 * 10^6 … lent !

Accélération

- On cherche à diminuer le nombre de calculs d'intersection
- On peut factoriser des calculs :
  - Regrouper des primitives
    - Très interressant lorsque la scène est fixe
  - Regrouper des rayons
    - Délicat, surtout pour les rayons secondaires

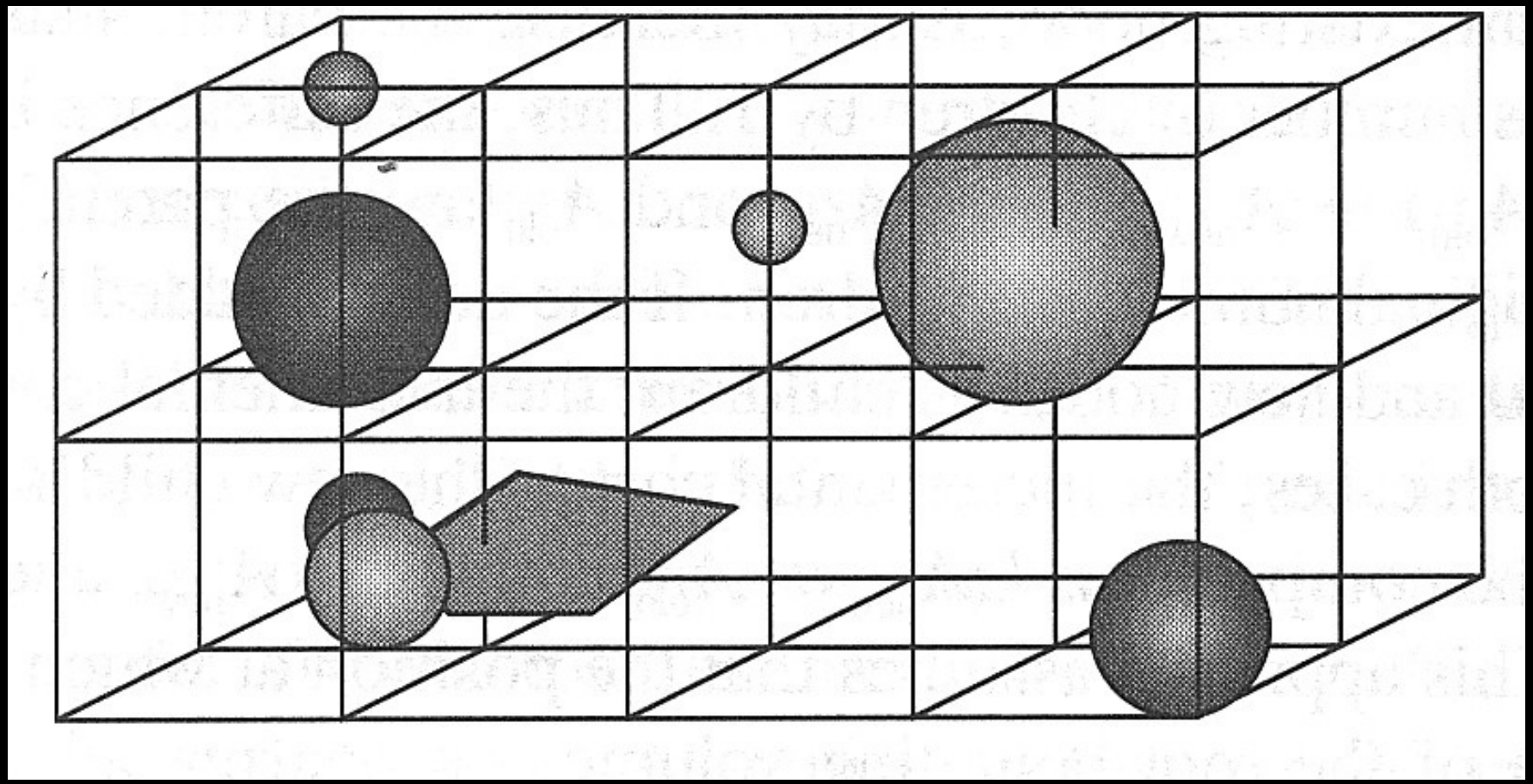
Structures sur la scène

- On va regrouper des primitives en utilisant une structure sur la scène
- Différentes possibilités :
  - Structures non hiérarchiques
    - Grille régulière
    - Boîtes englobantes
  - Structures hiérarchiques
    - Hiérarchie de boîtes englobantes
    - Kd-tree
    - Octree
    - Hiérarchie de grilles

Grilles

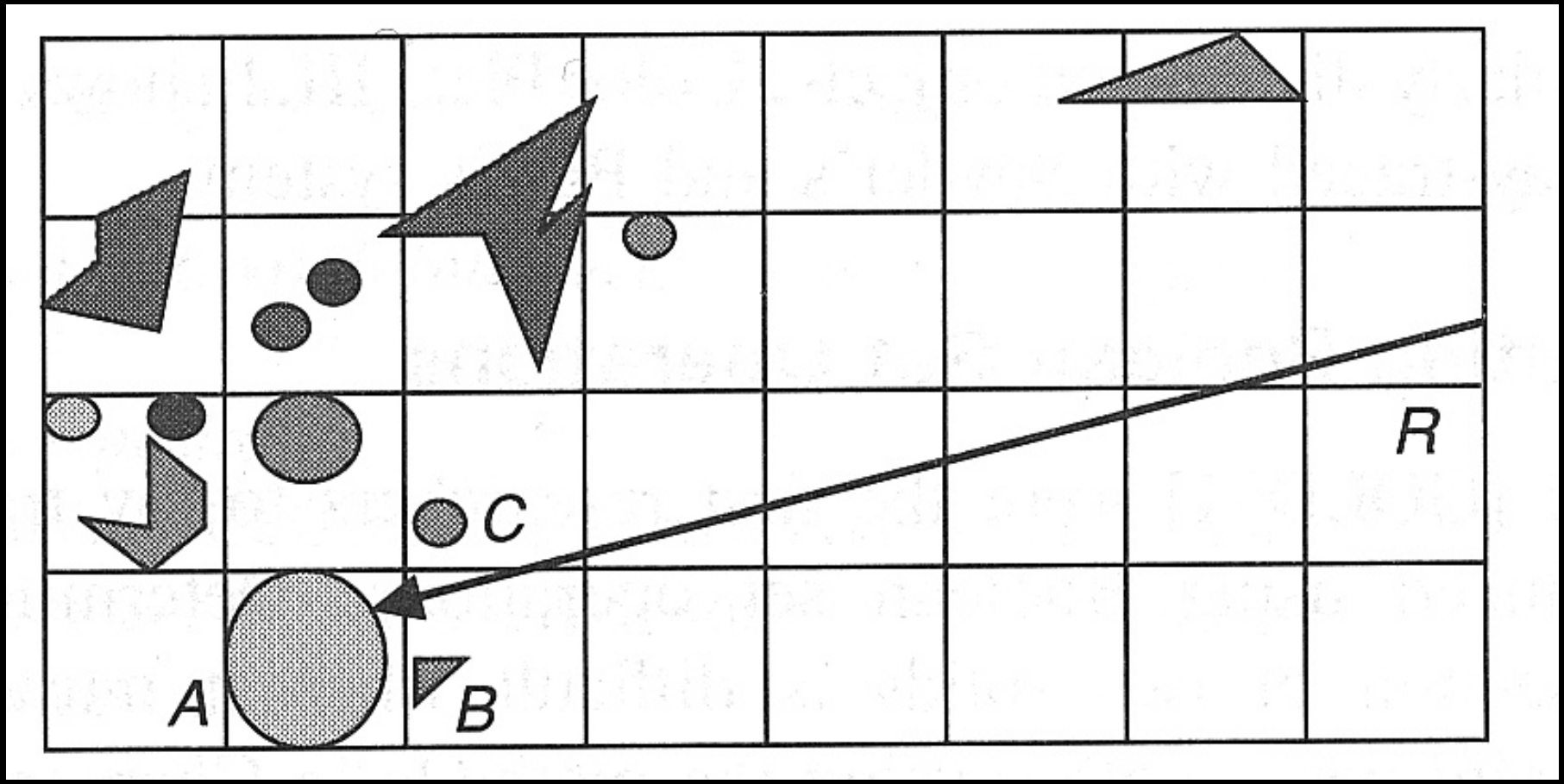
- Structure très simple
- Calculer la boîte englobante de la scène
- Diviser régulièrement
  - Souvent : Nombre de voxels = Nombre de primitives

Grilles



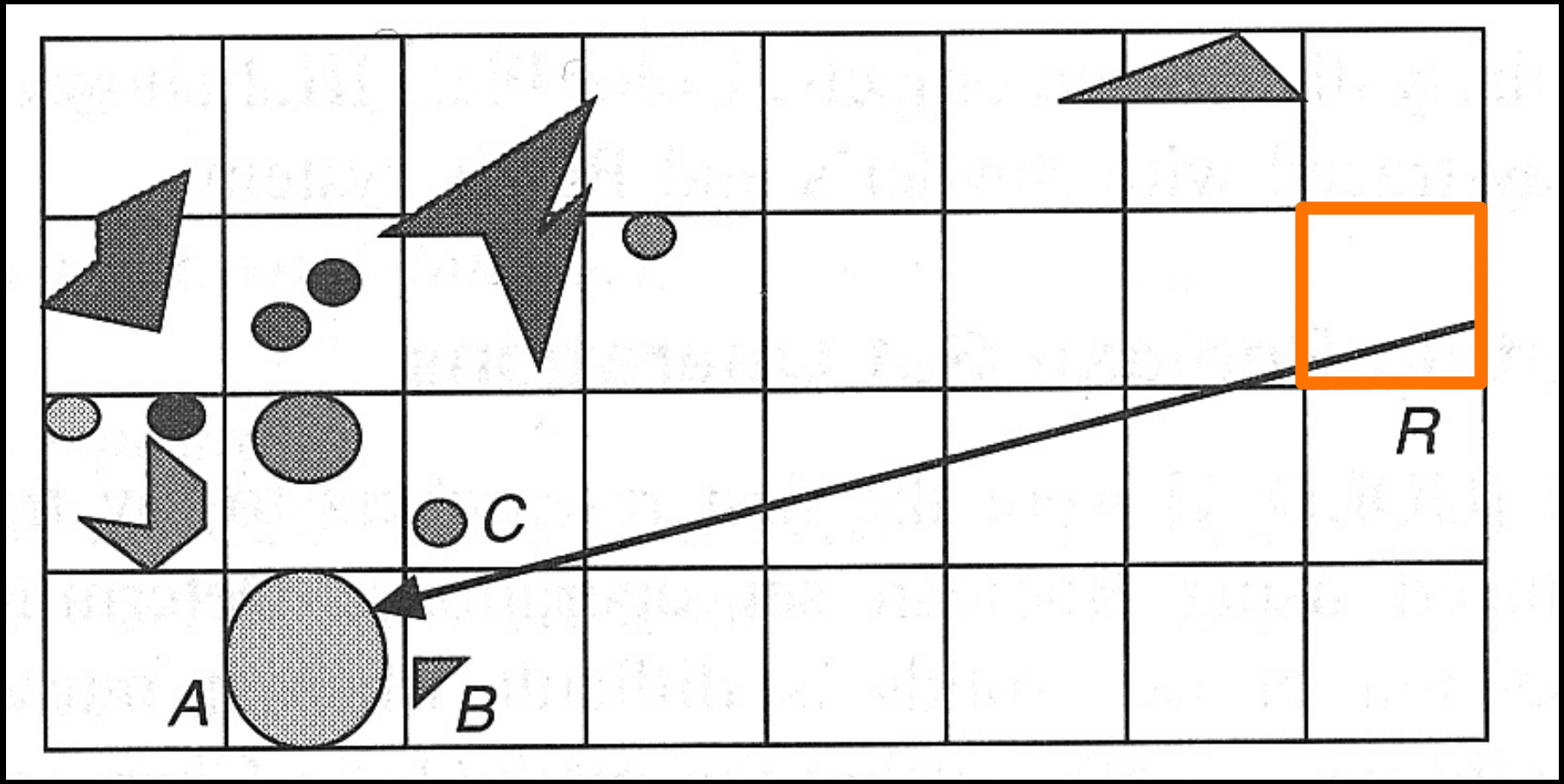
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



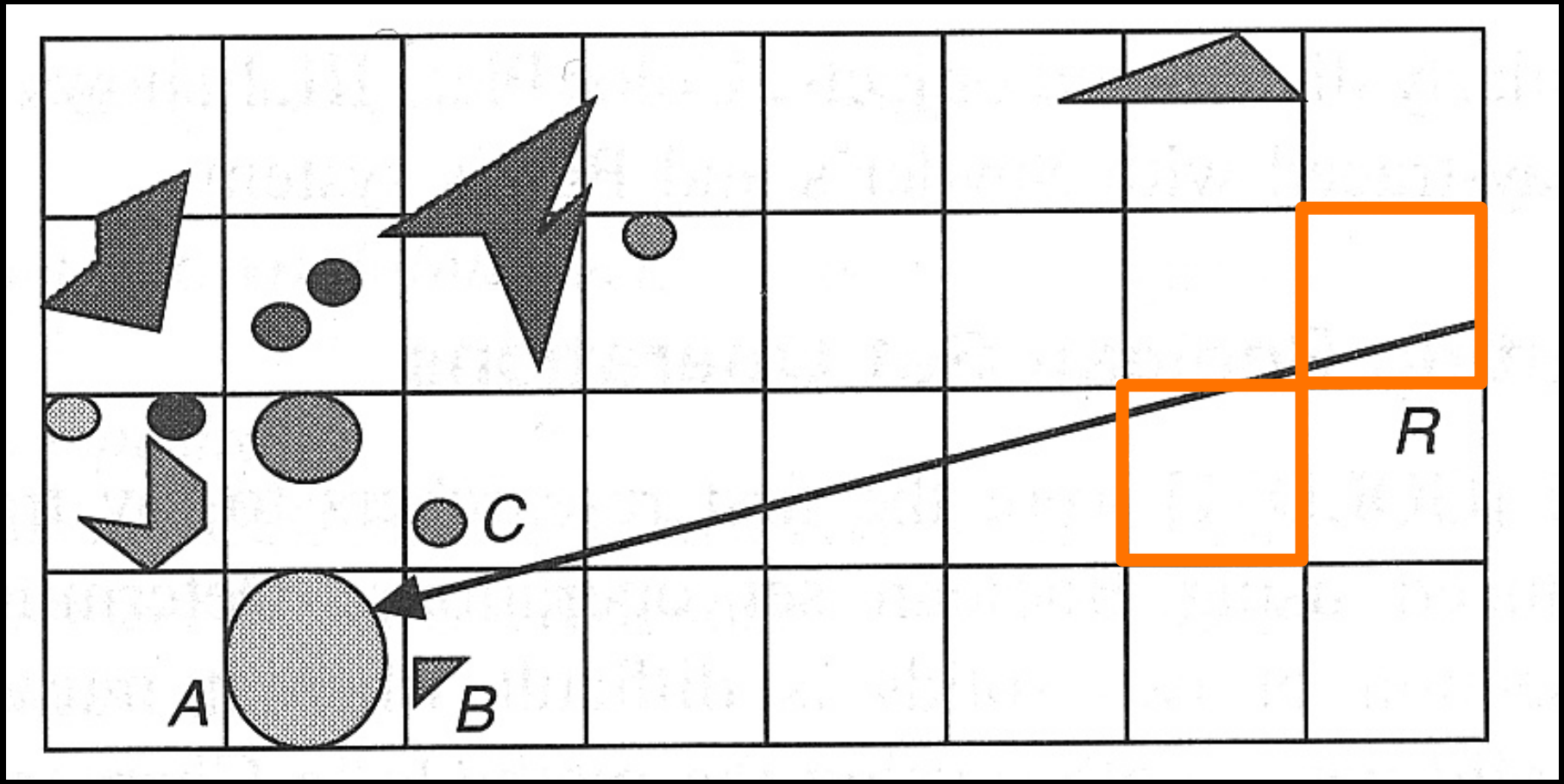
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



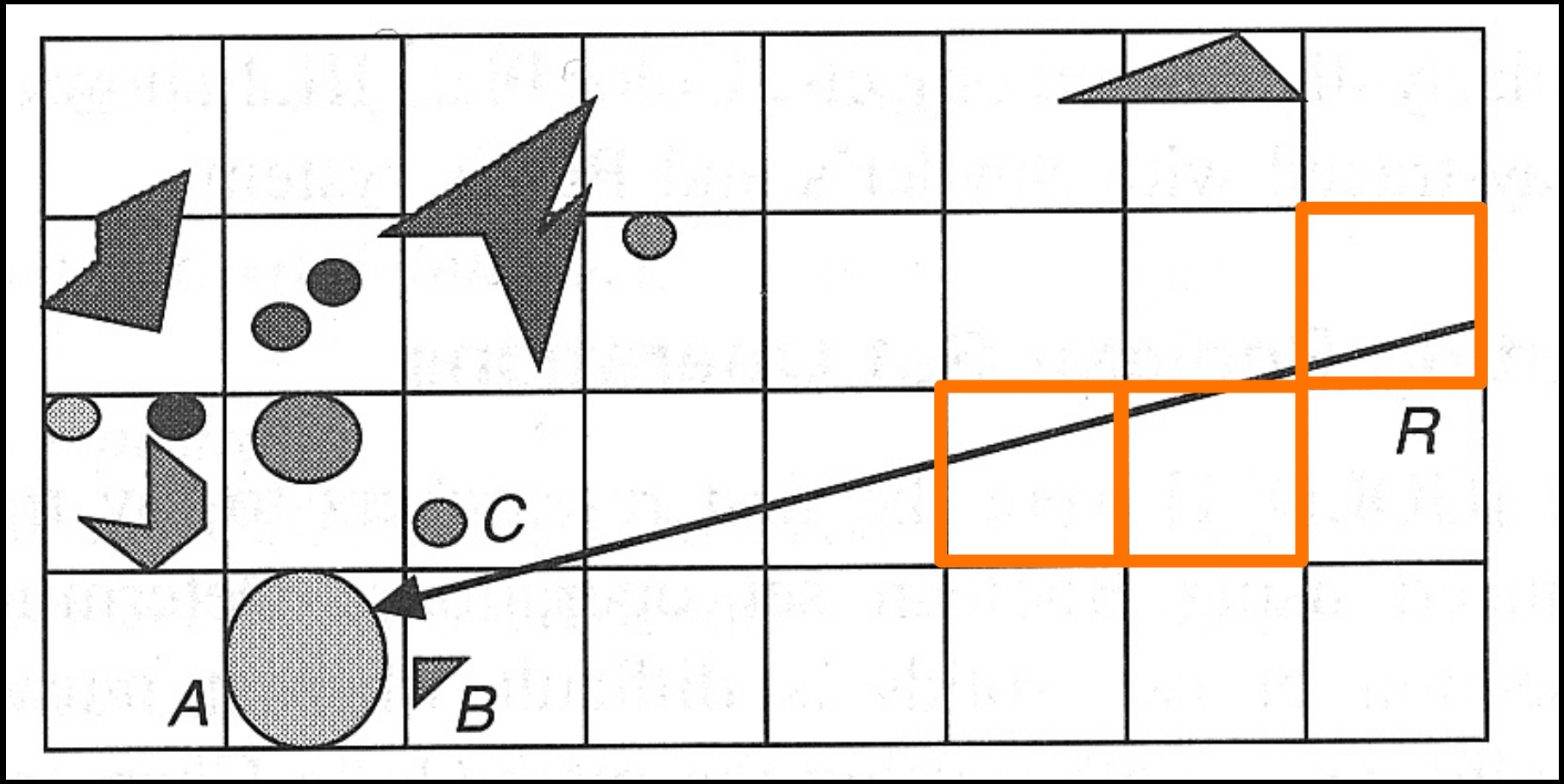
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



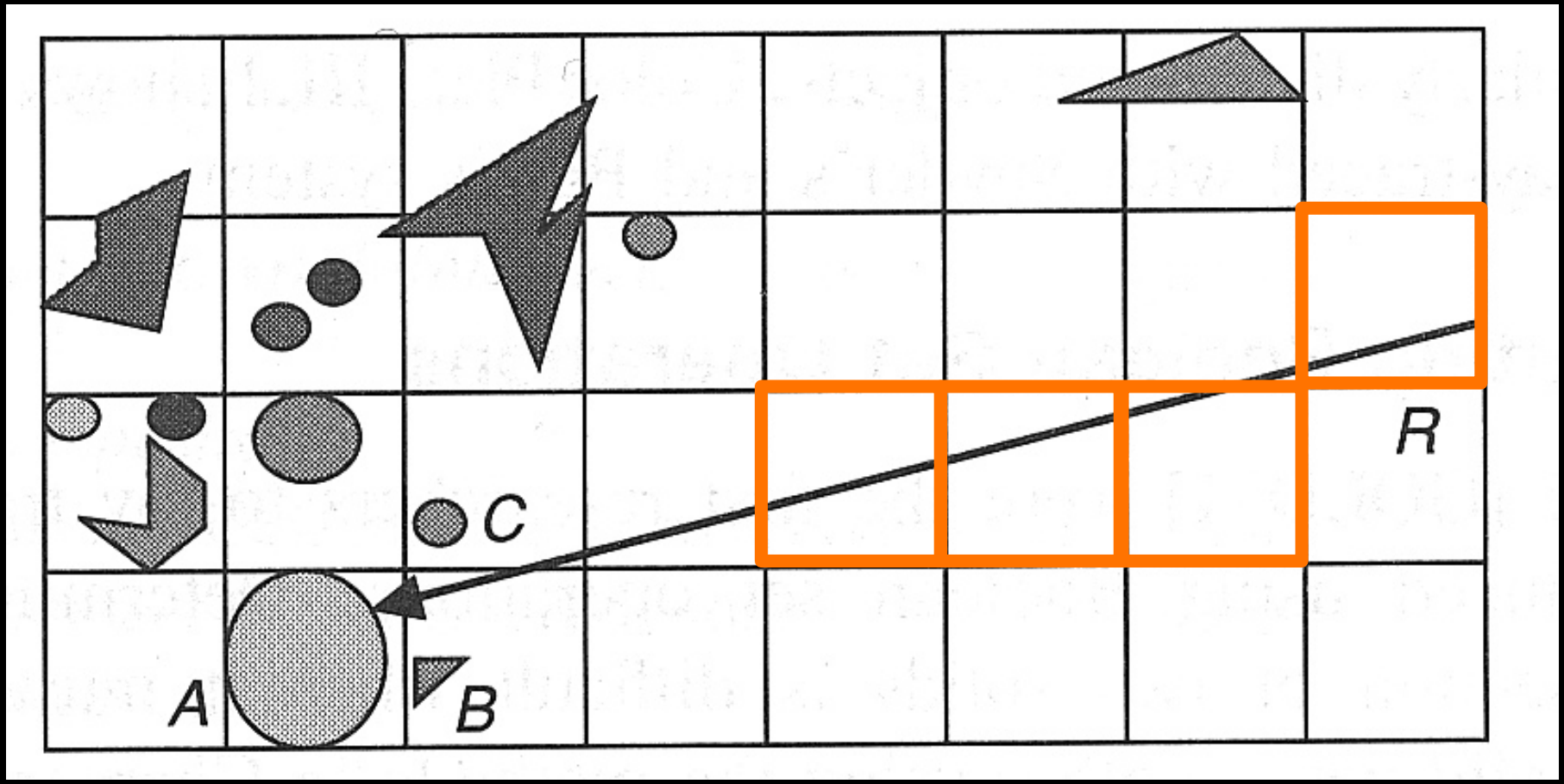
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



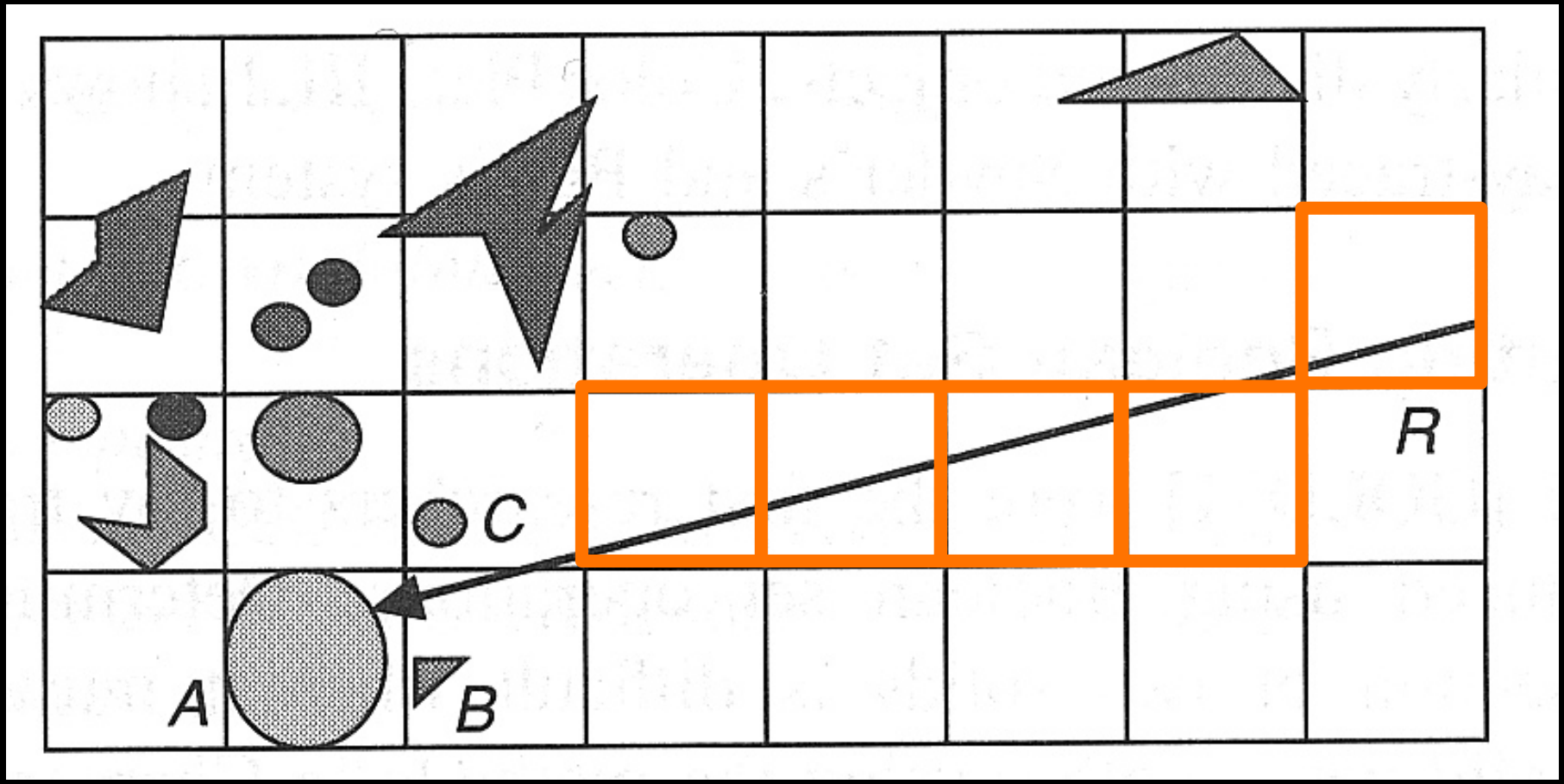
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



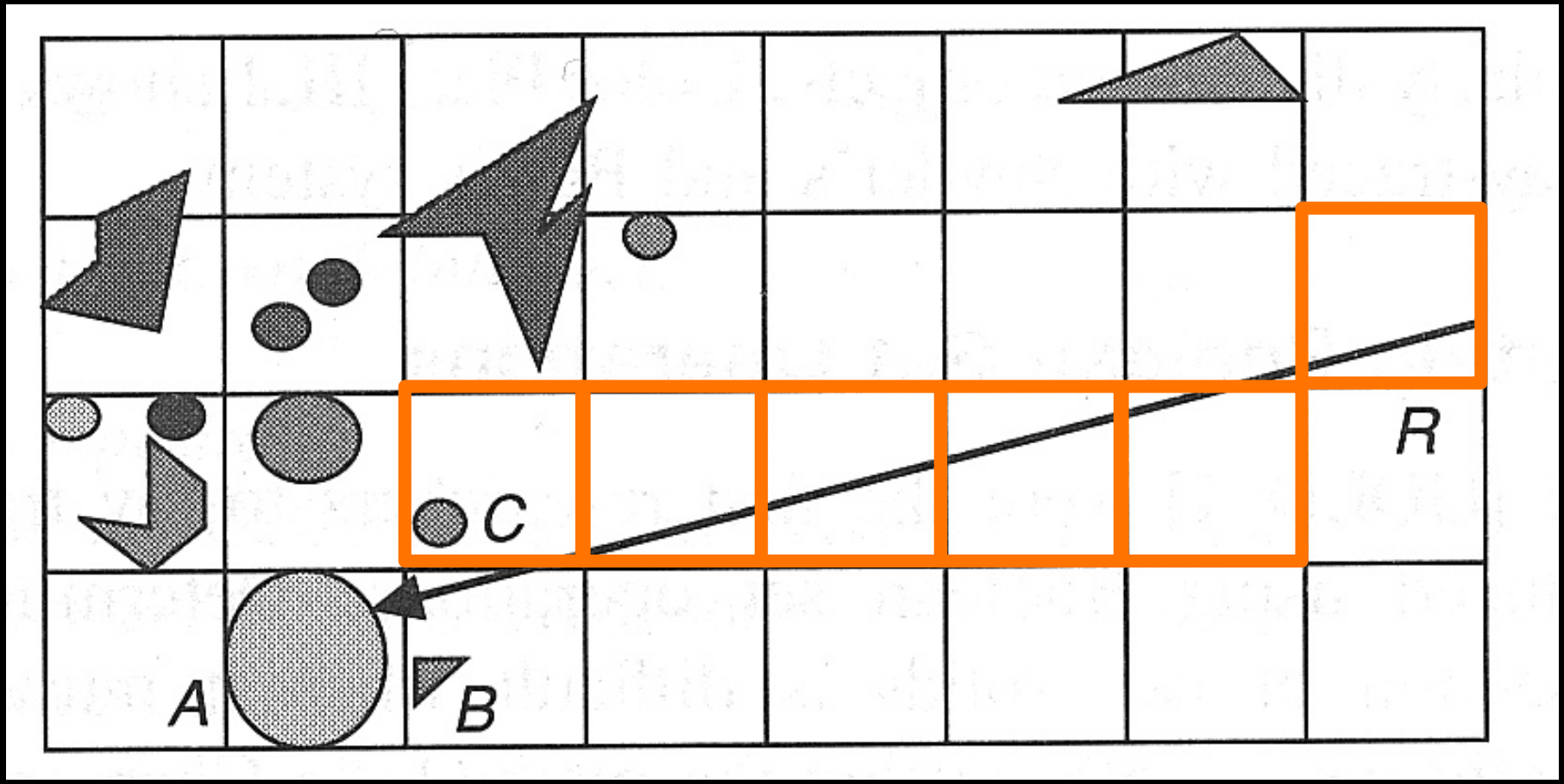
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



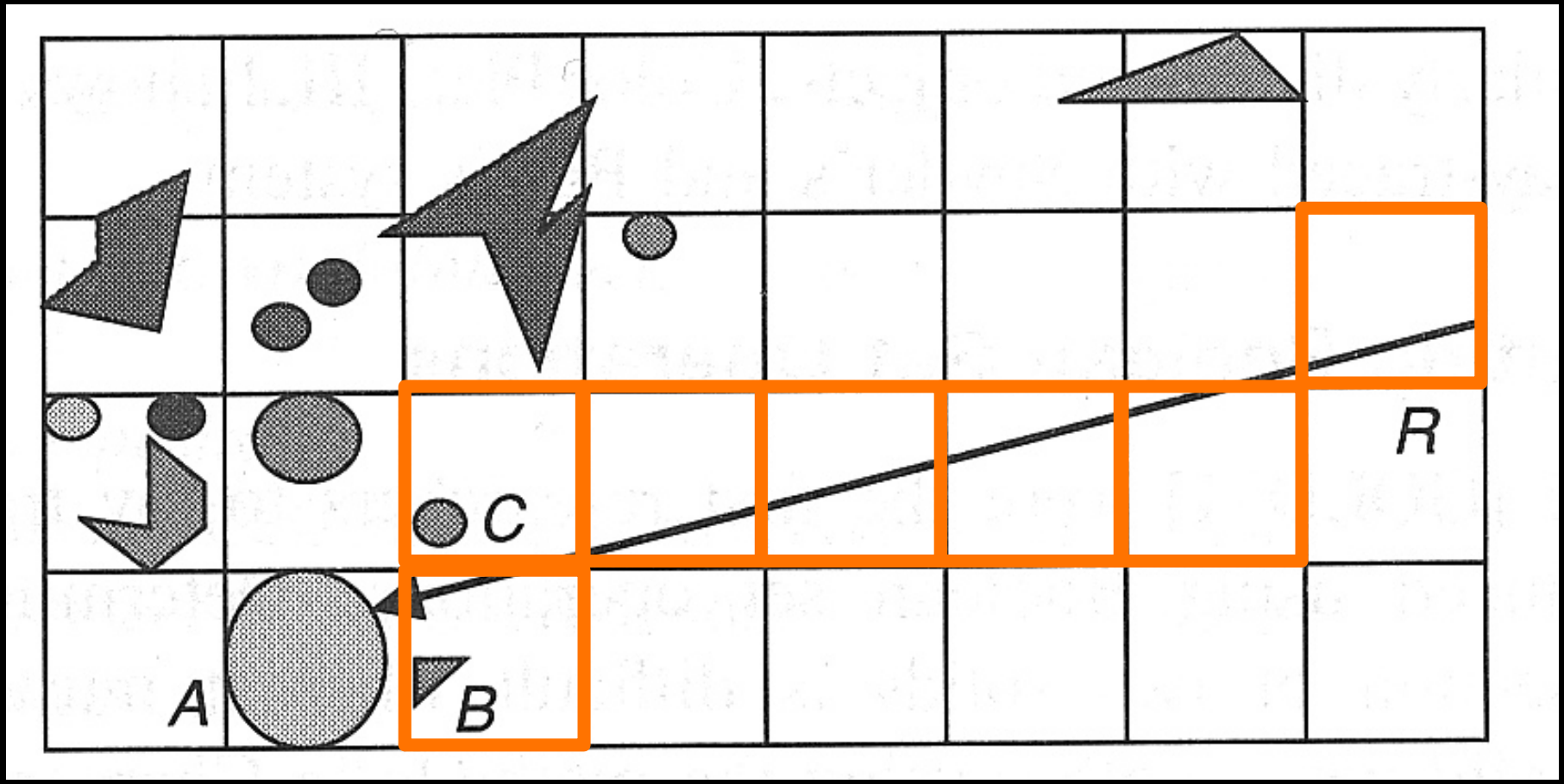
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



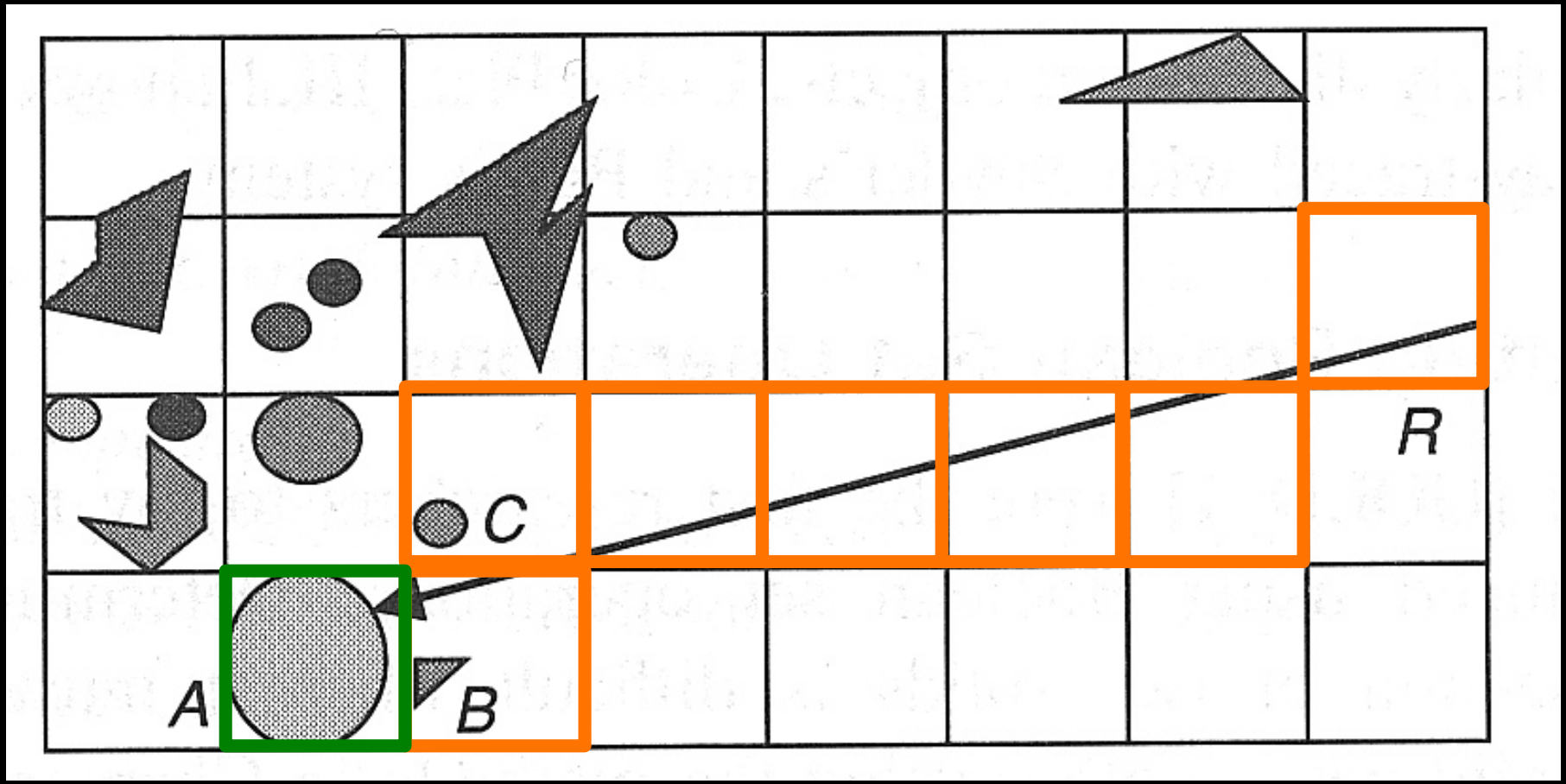
- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles



- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

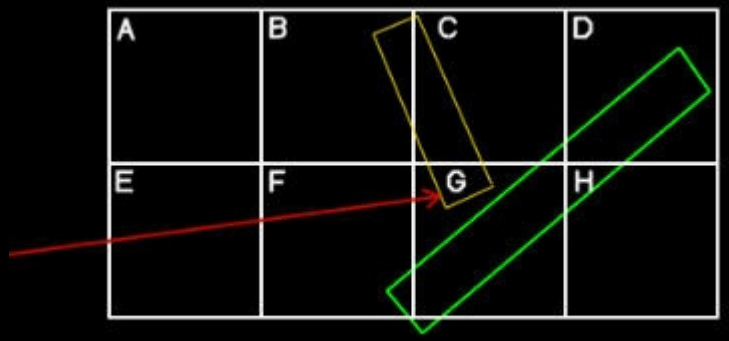
Grilles



- Chaque case contient une liste des objets qui l'intersectent.
- Chaque rayon progresse de case en case (algorithme 3D-DDA)
- On ne teste que les objets qui intersectent la case en cours

Grilles

- Principe des boîtes aux lettres (Mail-boxing)



Le rayon arrive en F :

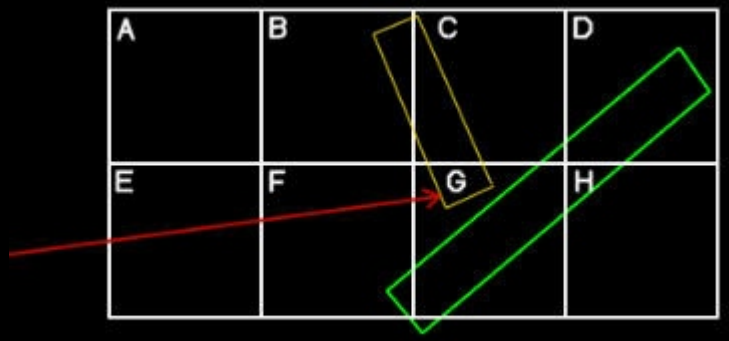
Le rectangle vert intersecte F

Le rayon est testé contre le rectangle vert

Il y a intersection, mais pas en F : on n'en tient pas compte

Grilles

- Principe des boîtes aux lettres (Mail-boxing)



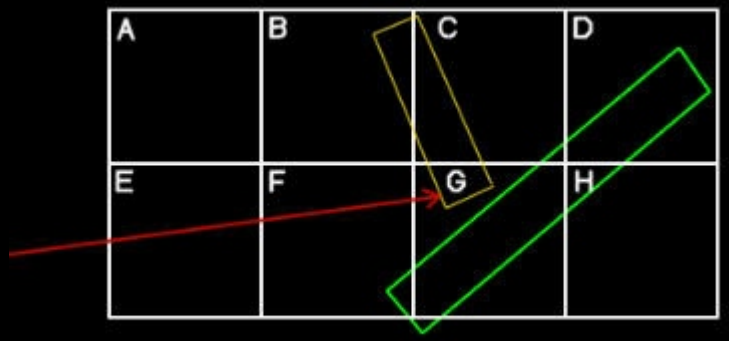
Le rayon arrive en G :

Le rectangle vert intersecte G

Le rayon est testé contre le rectangle vert (encore ?!)

Grilles

- Principe des boîtes aux lettres (Mail-boxing)

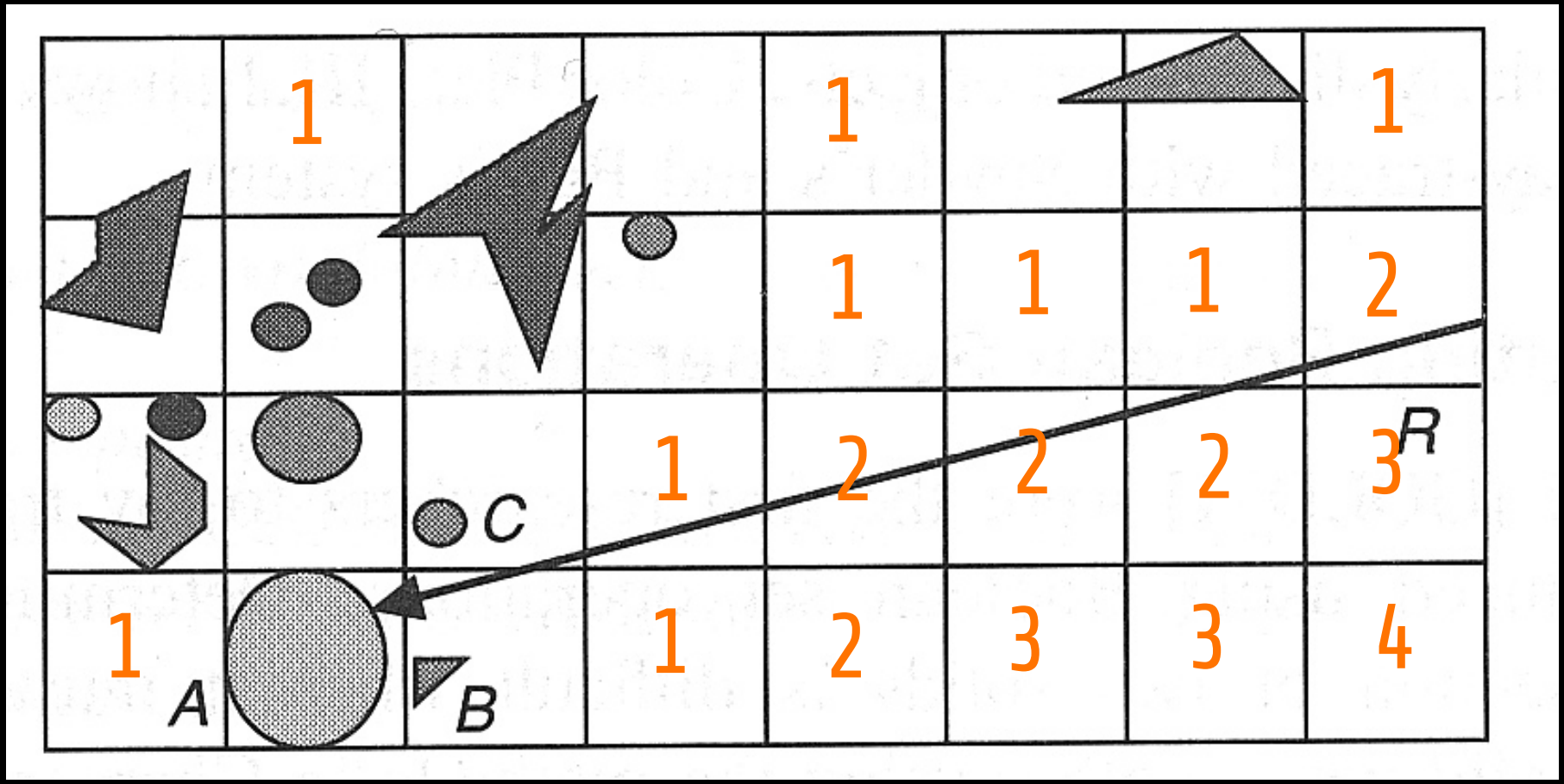


Pour éviter de faire deux fois le test :

on numérote les rayons

chaque objet maintient un cache (liste de taille limitée) des rayons qui l'ont intersecté

Optimisation : distance map

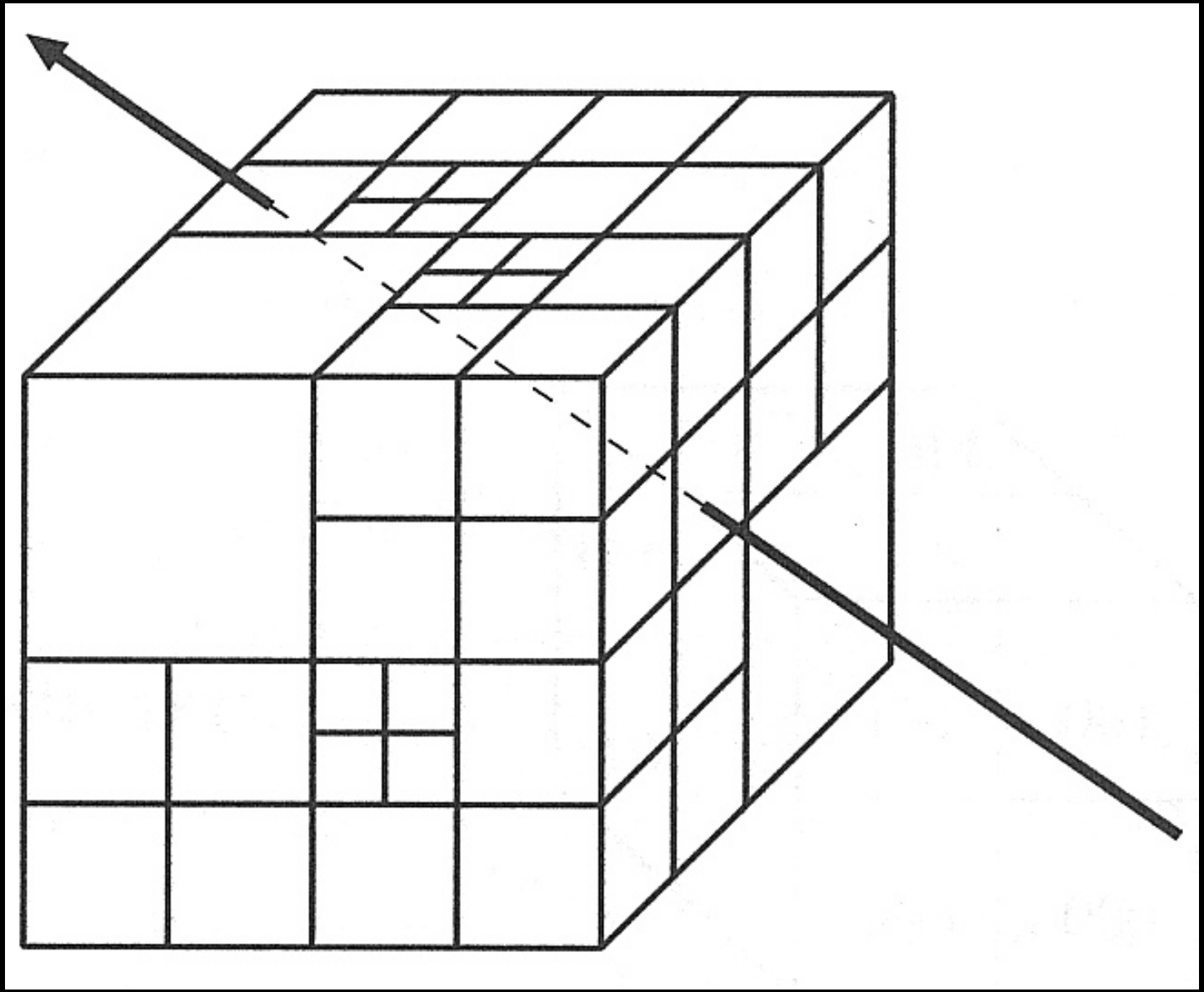


- Chaque case contient la distance à l'objet le plus proche
- Eventuellement, une pour chaque direction
- Attention quand la scène change

Grilles : le pour et le contre

- Pour :
  - Simple
  - Facile à mettre à jour quand la scène change
- Contre :
  - Mal adapté dans certains cas
    - « Teapot in a stadium »
  - La compléxité peut rester la même

Octree



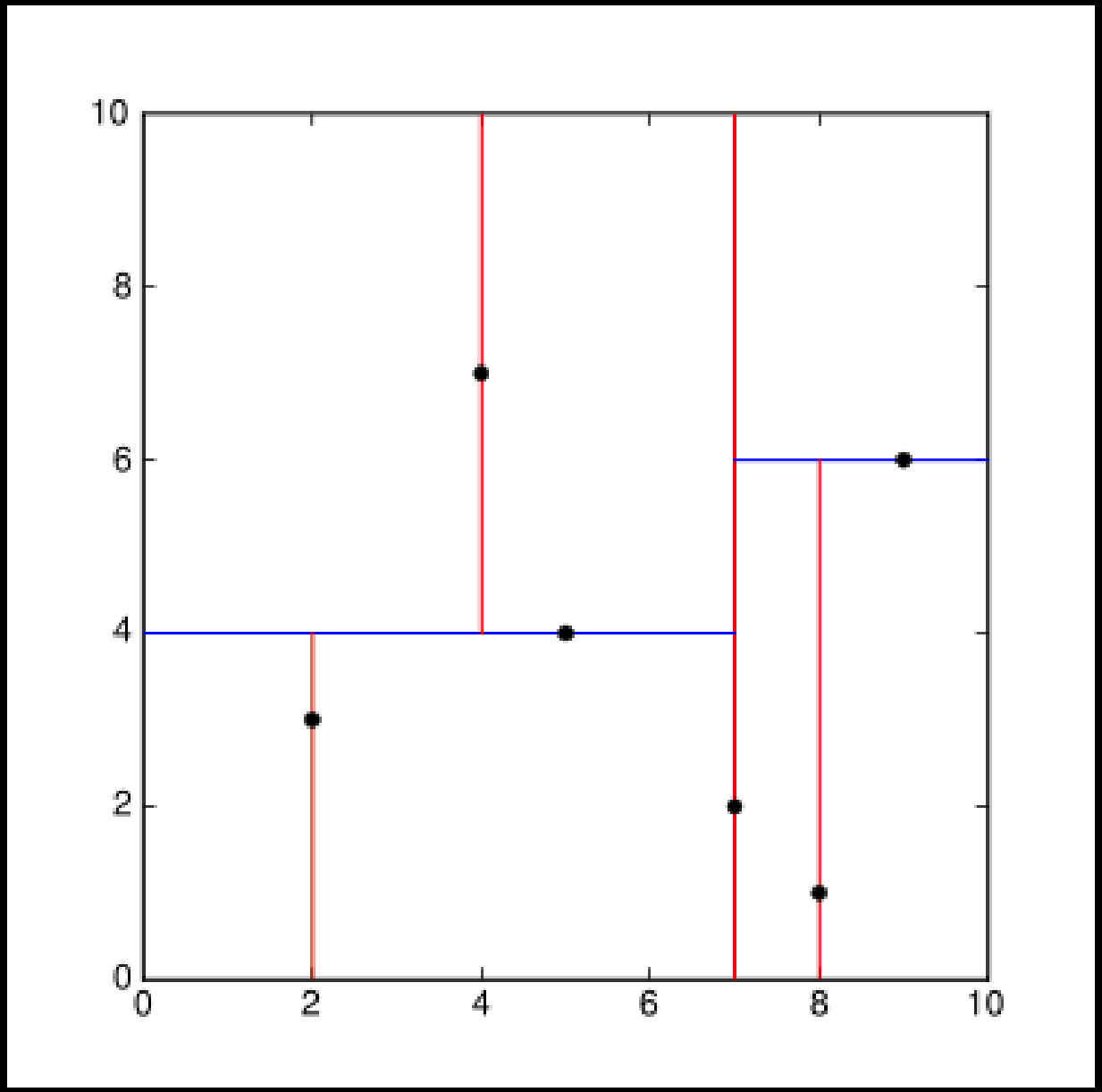
Octree

- Hiérarchie peu efficace
- Compliqué à parcourir
- Peu utilisé en pratique

Kd-tree

- Même principe que l'octree
  - Octree : noeuds divisés en 8 fils de même taille
  - Kd-tree : noeuds divisés en 2 fils, selon un des axes (x,y ou z), de tailles inégales
- Plus souple que l'octree
- Plus simple à parcourir

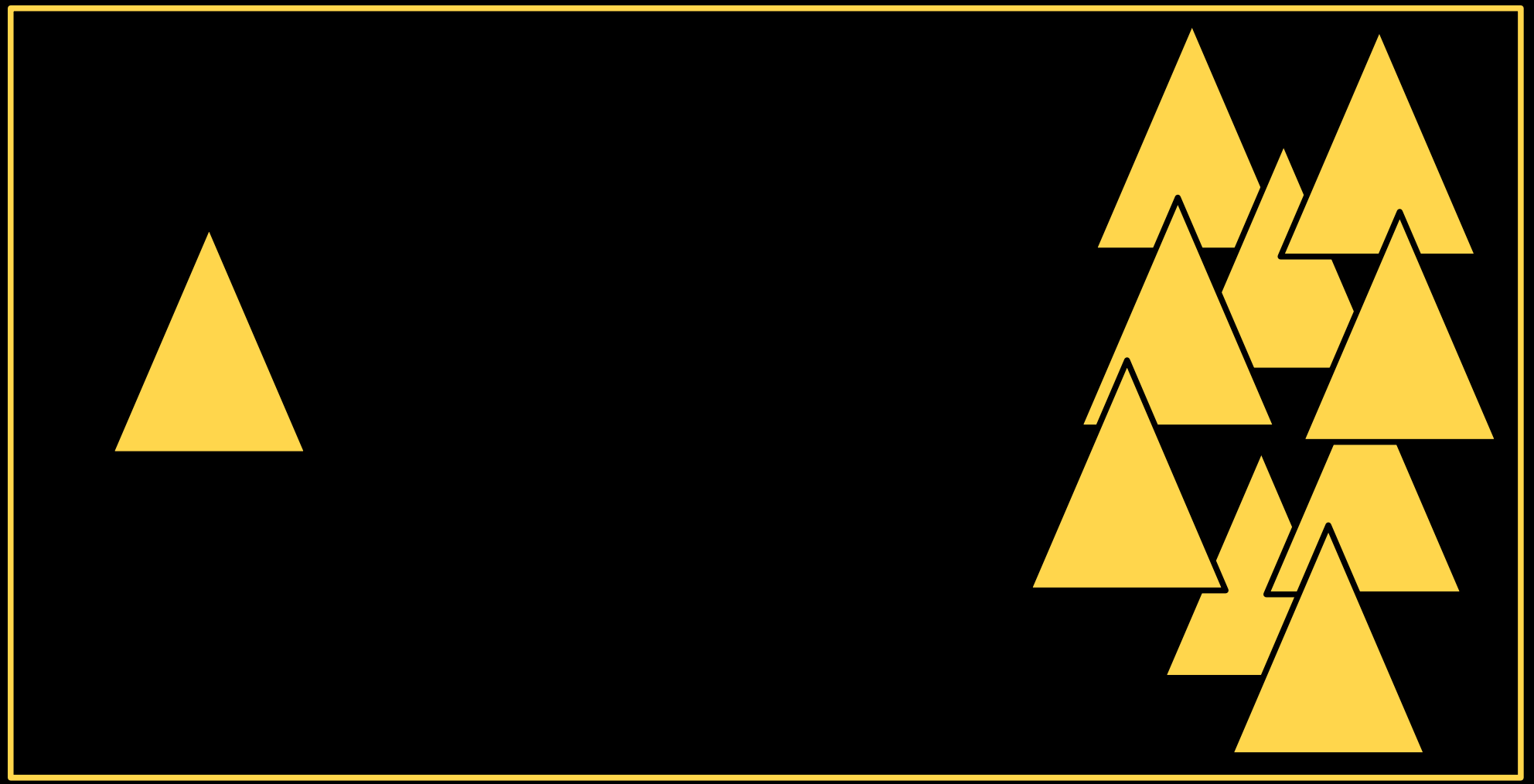
Kd-tree



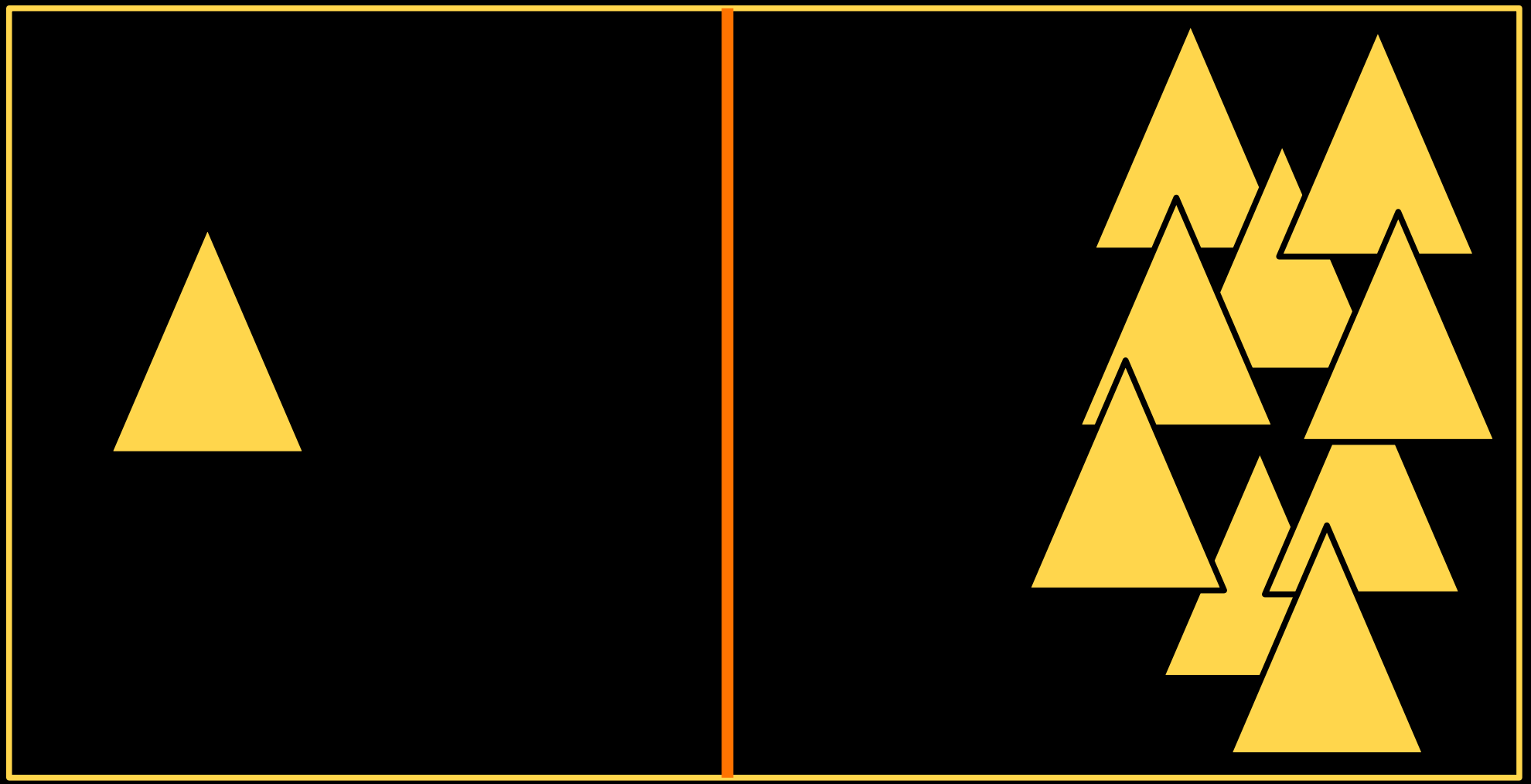
Kd-tree : construction efficace

- Cours SIGGRAPH 2005 de G. Stoll
- Techniques naïves :
  - Couper au milieu selon un des axes
  - Couper selon le médian (autant de primitives dans chaque fils)
- Technique efficace :
  - Surface Area Heuristic (SAH)
    - Optimisation d'une fonction de coût

Kd-tree : construction efficace

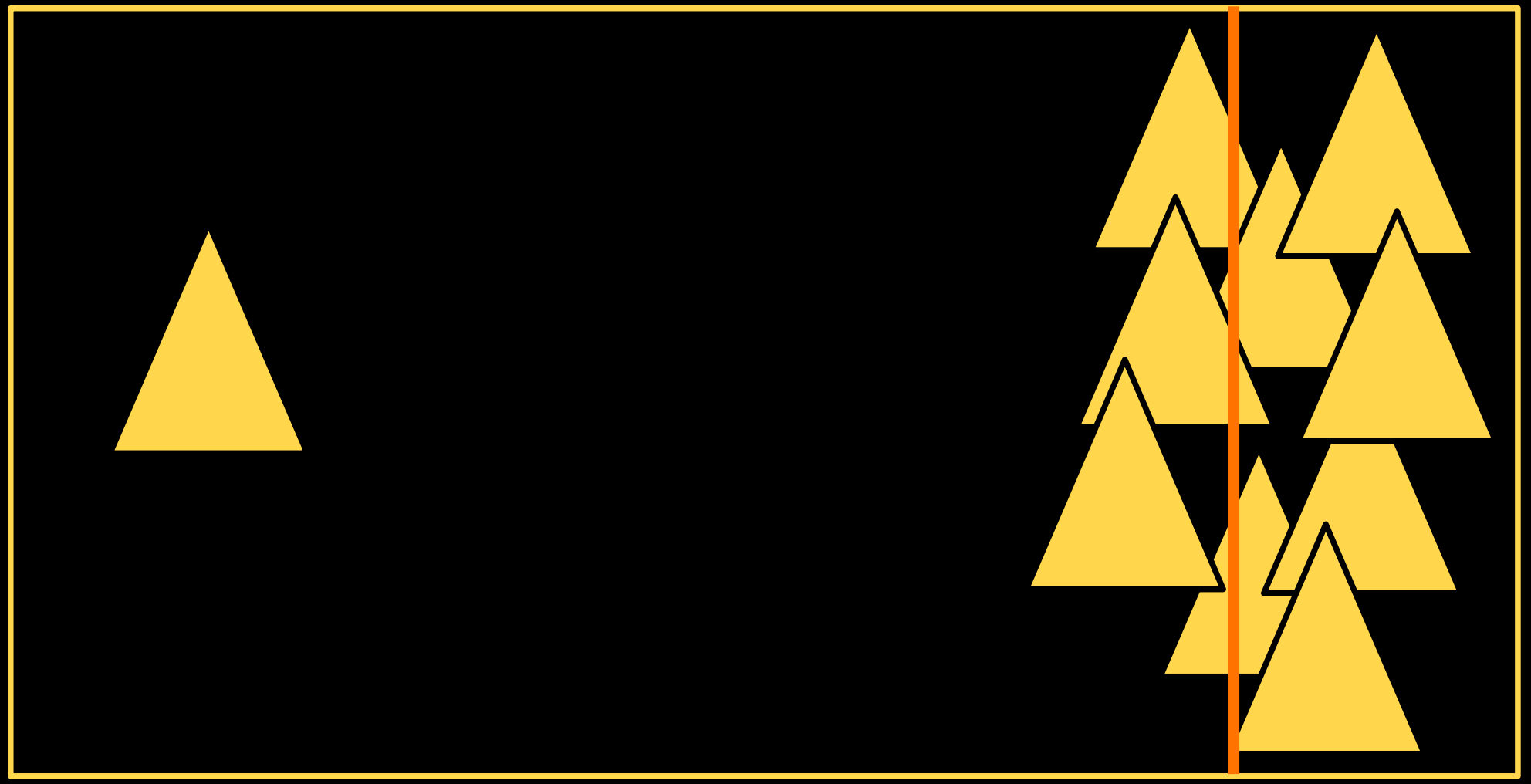


Kd-tree : construction efficace



- Milieu : cellules de même taille mais géométrie déséquilibrée

Kd-tree : construction efficace



- Médiane : géométrie bien répartie mais cellules mal proportionnés

Kd-tree : construction efficace

- SAH : équilibre entre taille des cellules et répartition de la géométrie

Kd-tree : le pour et le contre

- Pour :
  - Amélioration de la compléxité
    - O(nb Rayons * log(nb primitives))
  - Adaptatif (résout « teapot in a stadium »)
  - Rend le raytracing adapté aux très grosses scènes
- Contre :
  - Précalcul pour la construction
  - Scènes animées : mise à jour ou reconstruction nécessaire

Autres structures

- *A comparison of Acceleration Structures for GPU assisted Ray Tracing*, Master's Thesis, N.Thrane et L.O. Simonsen
- En particulier : hiérarchie de boîtes englobantes (BVH)

Meilleur choix ?

- Il n'y en a pas
- Chacune a ses points faibles
- En pratique pour les scènes fixes :
  - Kd-tree
  - BVH
- Pour les scènes animées :
  - BVH
  - Grilles

Aller encore 10x plus vite

- Regrouper des rayons
  - Packetisation
- Implémentation efficace
  - Utilisation mémoire
  - Caches
  - Parallélisme (multi threading, instructions SSE ...)
- Multi Level Ray Tracing :
  - [**Reshetov**. et al. SIGGRAPH 2005]

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Effets supplémentaires

- Si on a le temps, détail de :
  - Profondeur de champ
  - Motion blur
  - Caustiques
  - Reflections diffuses
  - Ombre douces
  - Coefficient de Fresnel

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Plan

- Algorithme de base
- Aliasing
- Accélération de l'algorithme
- Techniques avancées
- Conclusion

Synthèse

- Avantages :
  - Calcule que ce qui est utile (intersections et anti-aliasing)
  - Parallélisme
  - Grand choix de primitives
  - Adapté aux grosses scènes (log(n))
  - Effets réalistes (ombres, effets spéculaires ...)
- Désavantages :
  - Pas assez rapide pour le temps réel
  - Pas de hardware spécialisé
  - Scènes dynamiques mal supportées
  - Pas de vrai éclairage global (réflections diffuses, caustiques ...)

Problèmes ouverts

- Scènes dynamiques
- Hardware spécialisé

Fin